【新手学编程ABC丛书】

附赠DVD光盘

- 全书20个PPT教学资源
- 299分钟实例讲解视频
- 217段程序源代码

Java Web编程新手自学手册

扶松柏 等编著

进入职场就是现在！



机械工业出版社
CHINA MACHINE PRESS

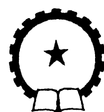
VISIT…

LANZAROTE Caliente.com

新手学编程 ABC 丛书

Java Web 编程新手自学手册

扶松柏　等编著



机 械 工 业 出 版 社

Java Web 是当今使用最为广泛的 Web 开发技术之一，一直在开发领域占据着重要地位。本书循序渐进、由浅入深地讲解了 Java Web 的核心技术，并通过具体实例的实现过程为读者展示了各个知识点的具体使用流程。全书共 20 章，分为 4 篇。第 1～6 章是基础篇，讲解了认识 Java，Java 语法，程序控制结构，数组，面向对象，异常处理等内容；第 7～12 章是提高篇，讲解了搭建 Java Web 开发环境，HTML 和 XML，JSP 指令和内置对象，标签和动作指令，Servlet 详解，JavaBean 组件技术等内容；第 13～15 章是数据库篇，讲解了数据库技术，使用 JDBC 连接数据库，数据库编程等内容，第 16～20 章是核心技术篇，讲解了 Struts2，Spring，Hibernate 以及它们的整合应用等内容。全书采用故事性、趣味性相结合的对话讲解方式，并穿插了学习技巧和职场点拨，引领读者全面掌握 Java。

本书不但适用于 Java Web 的初学者，也可供有一定 Java Web 基础的读者阅读。

图书在版编目（CIP）数据

Java Web 编程新手自学手册 / 扶松柏等编著. —北京：机械工业出版社，2012.10

（新手学编程 ABC 丛书）

ISBN 978-7-111-39087-9

Ⅰ. ①J… Ⅱ. ①扶… Ⅲ. ①JAVA 语言-程序设计 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2012）第 154245 号

机械工业出版社（北京市百万庄大街 22 号　邮政编码 100037）
策划编辑：丁　诚　杨　硕
责任编辑：杨　硕
责任印制：杨　曦

保定市中画美凯印刷有限公司印刷

2012 年 9 月第 1 版 • 第 1 次印刷
184mm×260mm • 33.75 印张 • 839 千字
0001－3500 册
标准书号：ISBN 978-7-111-39087-9
ISBN 978-7-89433-594-4（光盘）
定价：89.00 元（含 1CD）

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务	网络服务
社服务中心：（010）88361066	教材网：http://www.cmpedu.com
销售一部：（010）68326294	机工官网：http://www.cmpbook.com
销售二部：（010）88379649	机工官博：http://weibo.com/cmp1952
读者购书热线：（010）88379203	**封面无防伪标均为盗版**

从 书 序

从《杜拉拉升职记》谈起

近年来，职场小说备受青睐，李可老师的《杜拉拉升职记》更是受到广大读者的喜爱，还被搬上了银幕。对许多人来说，职场生涯占据了整个人生的很大一部分时间，怎样才能在职场中如鱼得水，是人们必须认真思考的重要问题。即将走上程序员岗位的读者朋友们，请自问是否已经对未来的职场生涯胸有成竹？

本丛书不但可以帮助初学者提前演练职场生活，而且对在职人员也有借鉴意义。技术方面的知识就不用再多说了，每一页所包含的内容都是作者多年来的技术结晶。阅读本丛书后，希望读者们不但能学到编程技术，而且能够提前体验到职场中的一些常见场景，为将来的职场生涯做一些准备。希望本书能为读者们解惑，也希望能激励读者们在这个行业继续奋斗下去，迎接大家的将是明媚的阳光。

程序员的各个阶段

按照掌握技术的熟练程度来划分程序员的不同阶段，可以大体分为 5 个阶段。

1）初学者：处在此阶段的可能是一名在校学生，可能是应届毕业生，也可能是准备从其他行业向编程行业转型的人员。其共同特点是刚开始学习编程知识，对每一个知识点都充满了好奇，对未来充满期望。

2）菜鸟：这里的菜鸟可能是技术菜鸟，也可能是职场菜鸟。特点是某项技术的基本知识已经学习完毕，但是没有经过项目的洗礼，尚需要实战演练来磨炼。这个阶段一般指处于试用期或者刚刚从事程序员工作的人员。

3）初级程序员：对项目开发的基本流程有了初步的认识，并通过工作实战演练了自己的技术。此阶段处于进一步与同事、上级、下级和客户交流的摸索阶段，也是逐渐融入职场的一个阶段。

4）高级程序员：开发经验丰富，技术扎实，对同事关系、上下级关系和客户关系已经如鱼得水，也是事业发展的瓶颈阶段。此阶段的程序员在职场中一般是软件高级工程师。

5）资深程序员：技术实力和人脉关系俱佳，一个项目任务能如探囊取物般轻松完成。但是也对自己的未来充满思索，想寻求待遇更好的职位，会考虑跳槽，也会考虑创业。

本丛书书目

根据综合考虑分析，本丛书首批书目如下。

C 语言编程新手自学手册

C#编程新手自学手册

Visual C++编程新手自学手册

Java Web 编程新手自学手册

Java 编程新手自学手册

PHP 编程新手自学手册

编程算法新手自学手册

技术菜鸟或职场菜鸟。特点是某项技术的基本知识已经学习完毕，但是没有经过项目的洗礼，尚需要实战演练来磨炼。



处在此阶段的可能是一名在校学生，可能是应届毕业生，也可能是准备从其他行业向编程行业转型的人员。其共同特点是刚开始学习编程知识，对每一个知识点都充满了好奇，对未来充满期望。



对项目开发的基本流程有了自己的认识，并通过工作实战演练了初步的技术。此阶段处于进一步与同事、上级、下级和客户交流的摸索阶段，也是融入职场的一个阶段。



开发经验丰富，技术扎实，对同事关系、上下级关系和客户关系已经如鱼得水，但也是事业进一步发展的瓶颈阶段。

技术实力和人脉关系俱佳，一个项目任务能如探囊取物般轻松完成。对自己的未来充满思索，想寻求待遇更好的职位。会考虑跳槽，也会考虑创业。



致读者

学习程序开发之路是充满挑战之路，也是充满乐趣之路，这条路没有捷径可走。梦想像《天龙八部》中虚竹那样轻松获得一甲子功力，是不现实的。读者们要想真正学好编程，需要付出辛苦的汗水。根据笔者的亲身体会，替读者总结出 3 条学习编程的建议。

（1）培养兴趣

无论做什么事情，只要有了兴趣，就喜欢花费时间去做它。只要喜欢感受那调试成功的喜悦，就说明已经对编程产生了兴趣。这种喜悦会使自己更加喜欢编程，会带来成就感。闲暇时刻建议多去专业编程论坛逛一逛，灌灌水。论坛里的朋友们不但能帮助自己解决问题，而且还能带来其他非技术性的收获。

（2）脚踏实地

欲速则不达，学编程切忌有浮躁的心态。很多初学者刚学会了基本语法知识，调试成功了几段代码，就迫不及待大声宣布："我精通 Java Web 了"。但是当遇到问题之后才发现，自己学到的只是九牛一毛。常说"书山有路勤为径，学海无涯苦作舟"，是很有道理的。

（3）多实践

程序开发很强调实践动手能力，所以实践就变得尤为重要。有前辈高人认为，学习编程的秘诀是"编程、编程、再编程，练习，练习，再练习"，笔者深表赞同。学编程不仅要多实践，而且要快实践。在看书的时候，不必等到完全理解了才动手，而是应该在看书的同时敲代码，程序运行的各种情况可以让自己更快、更牢固地掌握知识点。

我们的服务邮箱是 150649826@qq.com，读者在阅读本丛书时，如果发现错误或遇到问题，可以发送电子邮件及时与我们联系，我们会尽快给予答复。

丛书编委会

前　　言

Java Web 的重要性

Java Web 是企业级的网络系统，其中的“企业级”和“网络”（分布式）代表了 Java Web 开发的核心目标。在 Java Web 最新的规范中提供了进行企业级开发的框架组件 EJB，并同时提供了多种分布式服务开发技术方案，以及多种企业应用的业务解决方案。Java Web 基于强大的 Java 语言，从最初的 Servlet，发展到后来的 JSP，再到以 Struts、Hibernate、Spring 等为代表的框架，吸引了众多 Web 开发人员的目光。Java Web 开发在可重用性和安全性方面得到很好的保证，架构网站变得越来越方便和快捷。今天，已经有越来越多的人使用 Java Web 技术设计高效的电子商务和电子政务系统，开发各种中间交易系统，创建高水平的企业网站。

本书的特色

1）本书以一位初学者“小菜”的日记为主线，用一问一答的方式引出知识点，日记的格式记录了学习的历程，日记和学习、职场密切相关，也和知识点相关，希望读者读后能够受到启发。

2）本书遵循了理论加实践的写作模式，在每个知识点讲解完毕之后，都会用一个具体实例来演练知识点的用法。所有实例具有典型性和代表性。

3）本书揭示了一些职场规则和经验，循序渐进地向读者展示了学习、应聘、同事关系、客户关系、上下级关系、升职、跳槽和创业的经验和体会，给读者启示。

4）给读者以最大实惠：在配套光盘中不但有书中实例的源代码，而且配有 PPT 素材，还免费赠送给读者 9 个典型应用案例，并且为书中的项目案例都配备了详细的视频讲解。

本书的内容

Java Web 是当今使用最为广泛的 Web 开发技术之一，一直在开发领域占据着重要地位。本书循序渐进、由浅入深地讲解了 Java Web 的核心技术，并通过具体实例的实现过程为读者展示了各个知识点的具体使用流程。全书共 20 章，分为 4 篇。第 1～6 章是基础篇，讲解了认识 Java，Java 语法，程序控制结构，数组，面向对象，异常处理等内容；第 7～12 章是提高篇，讲解了搭建 Java Web 开发环境，HTML 和 XML，JSP 指令和内置对象，标签和动作指令，Servlet 详解，JavaBean 组件技术等内容；第 13～15 章是数据库篇，讲解了数据库技术，使用 JDBC 连接数据库，数据库编程等内容；第 16～20 章是核心技术篇，讲解了 Struts 2，Spring，Hibernate 以及它们的整合应用等内容。全书采用故事性、趣味性相结合的对话讲解方式，并穿插了学习技巧和职场点拨，引领读者全面掌握 Java。

本书不但适用于 Java Web 的初学者，也可供有一定 Java Web 基础的读者阅读。

本书的读者对象

初学编程的自学者　　编程爱好者

大中专院校的老师和学生　　相关培训机构的老师和学员

毕业设计的学生　　初中级程序开发人员

程序测试及维护人员　　参加实习的初级程序员

在职程序员　　资深程序员

致谢

本书的主要编写人员有谭贞军、陈强、张兴建、王梦、管西京、张子言、朱万林、李强、周秀、王孟、陈德春、周涛、刘海洋、关立勋、孟娜、王石磊、徐亮、张储、蒋凯、扶松柏、唐凯、焦甜甜、张斌、杨国华、杨絮、张玲玲。感谢作者们的辛勤汗水，也特别感谢机械工业出版社的所有工作人员，从他们身上我们学到了严谨的工作态度。

读者服务

在编写本书的过程中，我们始终本着科学、严谨的态度，力求精益求精，但错误、疏漏之处在所难免，敬请广大读者批评指正。我们的服务邮箱是 150649826@qq.com，读者在阅读本书时，如果发现错误或遇到问题，可以发送电子邮件及时与我们联系，我们会尽快答复。

感谢您购买本书，希望本书能成为您编程路上的领航者，祝您读书快乐！

编　者

目　　录

丛书序
前言

第一篇　基础篇

第 1 章　认识 Java……1
1.1　什么是 Java……2
1.1.1　Java 的发展历程……2
1.1.2　Java 语言的优势……2
1.1.3　Java 语言的魅力……3
1.2　搭建开发平台……4
1.2.1　下载 JDK……4
1.2.2　安装 JDK……6
1.2.3　配置 JDK……7
1.3　一个 Java 程序……9
1.3.1　编写一个 Java 程序……9
1.3.2　Java 程序的编译与运行……10
1.4　疑难问题解析……13
职场点拨——谈 Java 的地位……13
第 2 章　Java 语法……15
2.1　认识 Java 代码……15
2.2　标识符和注释……16
2.2.1　神奇的标识符……16
2.2.2　注释的力量……17
2.3　变量和常量……19
2.3.1　变量……19
2.3.2　常量……22
2.4　数据类型……22
2.4.1　了解简单数据类型……22
2.4.2　byte 型……22
2.4.3　整型……22
2.4.4　浮点数据类型……23
2.4.5　字符型……23
2.4.6　逻辑型……26
2.5　数据之间的转换……27
2.5.1　自动类型转换……27

2.5.2　强制类型转换 29
2.6　运算符与表达式 30
2.6.1　运算符 30
2.6.2　表达式 35
2.7　疑难问题解析 37
职场点拨——扎实的基础很重要 38
第 3 章　程序控制结构 39
3.1　认识 Java 语句 39
3.2　条件控制语句 40
3.2.1　if 语句 40
3.2.2　switch 语句 46
3.3　循环语句 49
3.3.1　for 循环语句 49
3.3.2　while 循环语句 51
3.3.3　do…while 循环语句 53
3.3.4　循环语句的嵌套 55
3.4　跳转语句 57
3.4.1　break 跳转语句 57
3.4.2　continue 语句 60
3.4.3　return 语句 62
3.5　疑难问题解析 64
职场点拨——看职业的发展前景 64
第 4 章　数组 66
4.1　简单的一维数组 66
4.1.1　声明一维数组 67
4.1.2　数组的创建和赋值 67
4.1.3　对数组进行初始化 70
4.2　二维数组 72
4.2.1　二维数组的声明 72
4.2.2　二维数组的创建 73
4.2.3　初始化二维数组 74
4.3　多维数组 77
4.3.1　声明和创建多维数组 77
4.3.2　初始化多维数组 77
4.4　操作数组 79
4.4.1　排序数组 80
4.4.2　对比数组 81
4.4.3　复制数组 82
4.4.4　填充数组 83

4.4.5 寻找数组 ……84
4.4.6 合并数组 ……85
4.5 疑难问题解析 ……86
职场点拨——学编程的误区 ……86
第 5 章 面向对象 ……88
5.1 类 ……89
5.1.1 使用 class 创建类 ……89
5.1.2 方法 ……89
5.1.3 特殊的方法——构造方法 ……90
5.2 类的访问 ……91
5.2.1 修饰符 ……91
5.2.2 this 的用法 ……92
5.2.3 使用类和对象 ……93
5.3 抽象类和抽象方法 ……94
5.4 包 ……96
5.4.1 定义包 ……96
5.4.2 导入包 ……98
5.5 类的继承 ……103
5.5.1 继承 ……103
5.5.2 对象的使用 ……105
5.5.3 对父类的访问 ……106
5.5.4 重载和重写 ……109
5.6 接口 ……111
5.6.1 接口的定义 ……111
5.6.2 接口常量和方法 ……112
5.6.3 多个接口的实现 ……113
5.7 疑难问题解析 ……114
职场点拨——我们是一个团队 ……115
第 6 章 异常处理 ……116
6.1 认识异常 ……116
6.2 什么是异常处理 ……119
6.2.1 异常处理基础 ……119
6.2.2 异常类 ……120
6.3 捕获异常 ……121
6.3.1 简单的异常捕获 ……121
6.3.2 多异常捕获 ……123
6.3.3 try…catch…finally 的组合 ……123
6.4 抛出异常 ……124
6.4.1 使用 throws 抛出异常 ……124

6.4.2　使用 throw 抛出异常 127
6.5　自定义异常的创建 129
6.6　异常处理的原则 131
6.7　杜绝异常处理的陋习 131
6.7.1　丢弃异常 132
6.7.2　不指定具体的异常 133
6.7.3　占用资源不释放 133
6.7.4　不说明异常的详细信息 133
6.7.5　过于庞大的 try 块 134
6.7.6　输出数据不完整 134
6.8　疑难问题解析 135
职场点拨——和上级相处之道 135

第二篇　提高篇

第 7 章　搭建 Java Web 开发环境 137
7.1　下载和安装 Tomcat 138
7.1.1　下载 Tomcat 138
7.1.2　安装 Tomcat 139
7.1.3　启动 Tomcat 140
7.1.4　设置 Tomcat 虚拟目录 141
7.1.5　一个简单的 Java Web 程序 143
7.2　下载和安装 MyEclipse 144
7.2.1　下载 MyEclipse 144
7.2.2　安装 MyEclipse 145
7.2.3　新建一个 MyEclipse 项目 147
7.2.4　部署开发环境 149
7.2.5　新建并运行一个 Java Web 页面 150
7.3　疑难问题解析 153
7.4　职场点拨——一份简历的思考 153
第 8 章　HTML 和 XML 155
8.1　认识 HTML 和 XML 155
8.2　理解 HTML 157
8.3　如何用 HTML 表现字体 157
8.3.1　标题就是这么酷 158
8.3.2　普通文字的大小、颜色和标记 159
8.3.3　将文字加粗、倾斜和加底线 159
8.4　整齐摆放文字 160
8.4.1　换行与不换行 160
8.4.2　文字的对齐 161

8.4.3 项目与列表 162
8.5 表格是表现数据的工具 163
8.5.1 表格的基本语法 163
8.5.2 跨多行或跨多列的表格 164
8.6 表单让网页实现了交互 166
8.6.1 表单容器<form> 166
8.6.2 单行文本框 167
8.6.3 单选按钮 168
8.6.4 多行文本和按钮 168
8.7 简单认识 CSS 样式和 JavaScript 169
8.7.1 编写和调用 CSS 样式 169
8.7.2 JavaScript 可以出现在网页的任何位置 171
8.8 XML 173
8.8.1 一个简单的 XML 173
8.8.2 XML 的其他元素 174
8.9 疑难问题解析 175
职场点拨——常见的面试错误 176
第 9 章 JSP 指令和内置对象 177
9.1 第一段 JSP 程序 177
9.2 JSP 概述 178
9.2.1 JSP 简介 178
9.2.2 JSP 的强势和弱势 179
9.3 认识 JSP 语言 179
9.3.1 编写第一个 JSP 程序 179
9.3.2 了解 JSP 基本语法 181
9.3.3 JSP 注释 181
9.3.4 JSP 的表达式 183
9.3.5 JSP 的声明语句 183
9.4 JSP 的指令标识 184
9.4.1 page 指令 184
9.4.2 include 指令 186
9.5 JSP 的内置对象 187
9.5.1 Request 对象 187
9.5.2 对象 Response 191
9.5.3 Session 对象 195
9.5.4 Out 对象 199
9.5.5 Application 对象 199
9.6 疑难问题解析 206
职场点拨——谈试用期 206

第 10 章　标签和动作指令 …… 207
10.1　认识标签和指令 …… 207
10.2　7 个动作指令 …… 209
10.2.1　forward 动作指令 …… 209
10.2.2　param 动作指令 …… 211
10.2.3　include 动作指令 …… 212
10.2.4　plugin 动作指令 …… 213
10.2.5　useBean,setProperty,getProperty 动作指令 …… 215
10.3　JSP 轻松定义自己的标签 …… 218
10.3.1　开发自定义标签类 …… 218
10.3.2　编写 TLD 文件 …… 218
10.3.3　使用标签库里的标签 …… 220
10.3.4　修改 web.xml …… 221
10.3.5　自定义标签的应用 …… 222
10.4　JSP 2.0 的新特性 …… 226
10.4.1　配置 JSP 属性 …… 226
10.4.2　JSP 的表达式 …… 228
10.5　疑难问题解析 …… 232
职场点拨——同事间的距离 …… 233
第 11 章　Servlet 详解 …… 234
11.1　认识 Servlet …… 234
11.2　Servlet 是什么 …… 239
11.2.1　Servlet 的技术介绍 …… 239
11.2.2　Servlet 技术的优越性 …… 239
11.2.3　Servlet 持久性 …… 240
11.3　开发一个 Servlet 程序 …… 240
11.3.1　编写一个 Servlet 源程序 …… 240
11.3.2　编译 Servlet 文件 …… 241
11.3.3　部署 Servlet 运行环境 …… 242
11.4　Servlet 的功能 …… 245
11.4.1　与 Servlet 配置相关的接口 …… 245
11.4.2　Servlet 编程接口 …… 246
11.4.3　Servlet 应用接口 …… 246
11.5　使用 HttpServlet 类封装请求 …… 248
11.5.1　Httpservlet 类 …… 248
11.5.2　请求与响应 …… 251
11.5.3　会话 …… 253
11.5.4　轻松将请求进行封装 …… 256
11.6　Servlet 过滤器 …… 260

11.6.1 过滤器的配置 260
11.6.2 过滤器的应用 261
11.7 Servlet 监听 263
11.8 疑难问题解析 263
职场点拨——寻兼职 264
第 12 章 JavaBean 组件技术 265
12.1 认识 JavaBean 265
12.2 JavaBean 基础 268
12.2.1 JavaBean 介绍 268
12.2.2 体验 JavaBean 268
12.2.3 在 JSP 中调用 JavaBean 270
12.3 设置 JavaBean 属性 273
12.3.1 Simple 属性 273
12.3.2 Indexed 属性 274
12.3.3 Bound 属性 275
12.3.4 Constrained 属性 276
12.4 JavaBean 方法 277
12.5 JavaBean 的作用范围 283
12.5.1 Page 作用域 283
12.5.2 Request 的作用域 283
12.5.3 Session 的作用域 283
12.5.4 Application 的作用域 283
12.6 JavaBean 的移除 284
12.7 疑难问题解析 288
职场点拨——兼职可靠吗？ 289

第三篇 数据库篇

第 13 章 数据库技术 290
13.1 认识 MySQL 数据库 290
13.2 操作 MySQL 294
13.2.1 MySQL 的安装 294
13.2.2 设置 MySQL 295
13.2.3 登录 MySQL 297
13.2.4 创建 MySQL 数据库 298
13.2.5 创建 MySQL 数据库表 298
13.2.6 输入记录 301
13.3 SQL Sever 可以这么简单 302
13.3.1 创建 SQL Sever 数据库 302
13.3.2 创建表 305

13.3.3 创建记录 ········ 307
13.4 疑难问题解析 ········ 308
职场点拨——想寻找更好的工作 ········ 309
第 14 章 使用 JDBC 连接数据库 ········ 310
14.1 连接数据库 ········ 310
14.2 什么是 JDBC ········ 312
14.2.1 JDBC API ········ 312
14.2.2 JDBC 对 B/S 和 C/S 模式的支持 ········ 313
14.2.3 JDBC 驱动类型 ········ 313
14.3 轻松连接 MySQL 数据库 ········ 314
14.3.1 下载 MySQL 的 JDBC 驱动 ········ 314
14.3.2 配置 MySQL 驱动 ········ 315
14.3.3 通过 Java 程序连接 MySQL ········ 315
14.3.4 在 Eclipse 中配置驱动 ········ 317
14.3.5 在 Eclipse 中测试是否连接成功 ········ 319
14.4 连接 SQL Sever 2000 ········ 320
14.4.1 安装 SQL Server 2000 的 JDBC 驱动 ········ 320
14.4.2 配置 SQL Server 2000 的 JDBC 驱动 ········ 321
14.4.3 连接数据库 ········ 322
14.4.4 在 Eclipse 中配置驱动 ········ 323
14.4.5 在 Eclipse 中测试是否连接成功 ········ 325
14.5 连接其他数据库的方法 ········ 325
14.6 JDBC 常用接口 ········ 326
14.6.1 DriverManager 类 ········ 326
14.6.2 Connection 接口 ········ 327
14.6.3 Statement 接口 ········ 327
14.6.4 PreparedStatement 接口 ········ 327
14.6.5 CallableStatement 接口 ········ 328
14.6.6 ResultSet 接口 ········ 329
14.7 疑难问题解析 ········ 329
职场点拨——准备创业 ········ 330
第 15 章 数据库编程 ········ 331
15.1 认识操作语句 ········ 331
15.2 数据定义语句 ········ 333
15.2.1 新建数据库和表 ········ 333
15.2.2 删除表 ········ 334
15.2.3 在已有表中进行编辑 ········ 334
15.3 数据查询语句 ········ 335
15.3.1 SELECT 语句 ········ 335

15.3.2 选择列表 ………… 336
15.3.3 FROM 语句 ………… 337
15.3.4 使用 Where 条件查询 ………… 337
15.3.5 联合查询 ………… 338
15.3.6 通过连接查询 ………… 338
15.4 数据库操作语句 ………… 340
15.4.1 用 INSERT 语句添加行 ………… 340
15.4.2 使用 UPDATE 语句修改行 ………… 342
15.4.3 删除语句 DELETE ………… 343
15.4.4 使用 ORDER BY 语句进行排序 ………… 343
15.4.5 使用索引 INDEX ………… 344
15.5 数据库控制语句 ………… 345
15.5.1 GRANT 语句 ………… 345
15.5.2 REVOKE 语句 ………… 346
15.6 数据库中常用函数 ………… 347
15.6.1 数学函数 ………… 347
15.6.2 聚合函数 ………… 347
15.6.3 字符串函数 ………… 348
15.6.4 日期和时间函数 ………… 348
15.6.5 加密函数 ………… 349
15.7 疑难问题解析 ………… 349
职场点拨——客户相处之道 ………… 350

第四篇　核心技术篇

第 16 章　Struts 2 基础 ………… 351
16.1 认识 Struts 2 ………… 351
16.2 Struts 2 的思想 ………… 354
16.2.1 MVC 思想 ………… 354
16.2.2 Struts MVC 思想 ………… 355
16.3 Struts 2 的获得与配置 ………… 356
16.3.1 获得 Struts 2 ………… 356
16.3.2 简单配置 Struts 2 ………… 357
16.3.3 在 MyEclipse 中配置 Strust 2 ………… 358
16.3.4 运行第一个 Struts 2 的程序 ………… 360
16.4 使用 Struts 2 标签库 ………… 365
16.4.1 Struts 2 标签库的组成 ………… 365
16.4.2 Struts 1 标签与 Struts 2 的区别 ………… 365
16.4.3 Struts 2 标签的使用 ………… 365
16.4.4 表单标签 ………… 368

16.4.5　控制标签 369
16.4.6　Struts 2 常用标签 370
16.5　编写 Action 371
16.5.1　使用 Action 动态调用 372
16.5.2　使用 Action 进行跳转 376
16.5.3　编写 Action 的经典法则 380
16.6　疑难问题解析 382
职场点拨——谈学习效率 382
第 17 章　Struts 2 架构 383
17.1　认识全新的 Struts 2 383
17.2　拦截器 388
17.2.1　什么是拦截器 388
17.2.2　配置拦截器 389
17.2.3　自定义拦截器 391
17.3　类型转换器 397
17.3.1　OGNL 的类型转换 397
17.3.2　自定义类型转换器 400
17.4　文件的上传和下载 403
17.4.1　文件的上传 403
17.4.2　文件的下载 408
17.5　疑难问题解析 411
职场点拨——教会徒弟饿死师傅 412
第 18 章　Spring 详解 413
18.1　认识 Spring 413
18.2　Spring 基础 414
18.2.1　为什么需要 Spring 414
18.2.2　Spring 将改变 Java EE 415
18.2.3　如何获得 Spring 415
18.2.4　部署 Spring 416
18.2.5　第一个 Spring 文件 419
18.3　Spring 的依赖注入 422
18.3.1　什么是依赖注入 422
18.3.2　设置注入 423
18.3.3　构造注入 427
18.4　Spring 的容器 428
18.4.1　Spring 常见的两个接口 428
18.4.2　使用 ApplicationContext 429
18.4.3　ApplicationContext 事件机制 432
18.5　Bean 对 Spring 的意义深远 435

18.5.1 Bean 的基本定义 435
18.5.2 容器中 Bean 的作用域 437
18.5.3 Spring 的 Bean 和 JavaBean 442
18.6 疑难问题解析 444
职场点拨——同事间的互补 445
第 19 章 Hibernate 详解 446
19.1 认识 Hibernate 446
19.1.1 Hibernate 常用接口介绍 447
19.1.2 Hibernate 下载和安装 448
19.1.3 书写第一个 Hibernate 程序 454
19.2 数据之间的类型的对应关系 466
19.3 Hibernate 的基本映射 472
19.3.1 主键生成器 472
19.3.2 映射集合属性 473
19.3.3 映射引用属性 476
19.4 数据库对象映射 479
19.4.1 如何将普通表转换成持久数据库表 479
19.4.2 创建 Hibernate 持久对象 480
19.4.3 描述对象之间的关系 483
19.4.4 使用映射关系 485
19.5 疑难问题解析 489
职场点拨——谈升职 490
第 20 章 Struts 2、Spring 和 Hibernate 整合应用 493
20.1 效果展示 493
20.2 搭建一个整合环境 494
20.2.1 Tomcat 虚拟目录与 MyEclipse 一致 494
20.2.2 先搭建 Hibernate 499
20.2.3 再搭建 Spring 501
20.2.4 Struts 2 的添加 504
20.3 简单的工程 506
20.3.1 新建数据库 506
20.3.2 配置 Hibernate、Spring 里的信息 507
20.3.3 新建几个页面 508
20.3.4 提交页面的处理 511
20.3.5 将用户信息显示出来 515
20.3.6 使用 struts 2 校验框架对表单进行校验 517
20.3.7 删除用户信息 518
20.3.8 更改用户信息 519

第一篇　基　础　篇

第1章　认识Java

Java Web技术是Java语言体系的一个分支，是指用Java技术开发Web应用。因为需要使用Java语言开发Java Web项目，所以在学习Java Web之前，一定要先学会Java语言的基本知识，特别是语法方面的知识。Java 语言有跨平台、一次编译多次运行、安全等特点，深受广大程序员的爱好和企业追捧。它是怎么发展到今天的？它的优势到底在哪里？怎么样才能在自己的电脑上编译和运行一个 Java 程序？在本章将一一为大家解答这些问题。通过本章能学到如下知识。

- 什么是Java。
- 编写一个Java程序。
- 职场点拨——谈Java的地位。

背景介绍

小菜，21岁，一名即将毕业的大四学生，正准备开始自学Java Web。

Wisdom，小菜的表哥，一名资深软件工程师。

2008年X月X日，天气阴

今天老师建议我们选修 Java Web。在过去的三年里，我学过了 C 语言和 C++，现在开始学习Java Web，对于我以后的职场生涯来说，不知道还有必要吗？



一问一答

小菜：“我已经学过了 C 语言和 C++，快毕业了才开始学 Java Web，还有必要吗？”

Wisdom：“当然有必要。Java Web属于Java体系，Java是当前最主流的编程语言之一。在计算机技术快速发展的今天，Java 广泛应用于各个领域，例如 Web 系统、桌面程序和手机应用，其中Java Web已经成为当前动态网站开发的主流技术之一。”

小菜：“看来Java很强大啊！”

Wisdom：“是的，建议你阅读本章最后的“谈 Java 的地位”，你会得到一个更加明确的答案。”

1.1 什么是 Java

Java 语言的应用范围比较广，可以开发 Web 项目、桌面项目和移动嵌入式项目，本书所要讲解的 Java Web 就是使用 Java 语言开发动态 Web 的一门技术。在本节的内容中，将简要讲解 Java 语言的基本知识，为读者的学习打下基础。

1.1.1 Java 的发展历程

Java 诞生于 1995 年 5 月，是由Sun Microsystems公司推出的程序设计语言（以下简称 Java 语言）和 Java 平台的总称。用 Java 实现的 HotJava，浏览器（支持 Java applet），显示了 Java 的强大魅力，例如跨平台、动态的Web以及Internet计算等。从此，Java 被广泛接受并推动了 Web 的迅速发展，常用的浏览器现在均支持 Java applet，并且 Java 技术还一直在不断地更新中。

Java 平台由 Java 虚拟机（Java Virtual Machine）和 Java 应用编程接口（Application Programming Interface，API）构成。Java 应用编程接口为 Java 应用提供了一个独立于操作系统的标准接口，这个接口可分为基本部分和扩展部分。当在硬件或操作系统平台上安装 Java 平台后，即可运行 Java 应用程序。现在 Java 平台已经嵌入了几乎所有的操作系统中，并且 Java 程序只编译一次就可以在各种系统中运行。Java 应用编程接口已经从 Java 1.1x 版发展到 Java 1.2 版，并继续发展。目前最常用的 Java 平台基于 Java 1.6，最近版本为 Java 1.7。

Java 分为如下三个体系。

1）JavaSE：Java2 Platform Standard Edition，Java 平台标准版。

2）JavaEE：Java2 Platform Enterprise Edition，Java 平台企业版。

3）JavaME：Java2 Platform Micro Edition，Java 平台微型版。

2009 年 4 月 20 日，Oracle（甲骨文）收购 Sun，从此以后，Java 归甲骨文公司所有。

1.1.2 Java 语言的优势

Java 语言具有优秀的特点，在互联网位置上占有重要的位置，其特点介绍如下。

1）Java 语言是简单的：Java 语言的语法与 C 语言和 C++语言类似，所以大多数程序员能够很容易地学习和使用 Java。在另一方面，Java 丢弃了 C++ 中很少使用的、很难理解的、令人迷惑的特性，例如操作符重载、多继承、自动的强制类型转换等。最大的区别是 Java 语言中不再使用指针，并提供了自动废料收集功能，这样程序员再也不必为内存管理而担忧了。

2）Java 语言是一门面向对象的语言：在 Java 中提供了类、接口和继承等概念，为了简单起见，只支持类之间的单继承，但是支持接口之间的多继承，并支持类与接口之间的实现机制（关键字为 implements）。Java 语言全面支持动态绑定，而 C++ 语言只对虚函数使用动态绑定。由此可见，Java 语言是一门纯正的面向对象程序设计语言。

3）Java 语言是分布式的：Java 功能强大，支持 Internet 方面的应用开发。通过基本 Java 应用编程接口中的网络应用编程接口（Java.NET），实现了用于网络应用编程的类库，包括 URL、URLConnection、Socket 和 ServerSocket 等。Java 的 RMI(远程方法激活)机制也是开发分布式应用的重要手段。

4）Java 语言是健壮的：Java 所具备的强类型机制、异常处理和废料自动收集等功能，

是 Java 程序健壮性的重要保证。并且 Java 丢弃了指针，使得效率更高。另外，Java 的安全检查机制使得 Java 更具健壮性。

5）Java 语言是安全的：Java 通常被用在网络环境中，所以强大的安全机制对于 Java 来说非常重要。在 Java 中，使用安全机制可以防止恶意代码的攻击。除了 Java 语言的许多安全特性以外，还有一个网络下载的类 ClassLoader，这是一个具有安全防范机制的类，例如分配不同的名字空间以防替代本地的同名类、字节代码检查，并提供安全管理机制（类 SecurityManager）实现安全设置。

6）Java 语言是体系结构中立的：Java 程序（扩展名为.java 的文件）在 Java 平台上被编译为体系结构中立的字节码格式（扩展名为.class 的文件），然后可以在实现这个 Java 平台的任何系统中运行。这种途径适合于异构的网络环境和软件的分发。

7）Java 语言的可移植性强：可移植性是指在不同平台之间能够相互移植，这种可移植性来源于体系结构的中立性。并且，Java 还严格规定了各个基本数据类型的长度。另外，Java 系统本身也具有很强的可移植性。

8）Java 语言是解释型的：Java 程序在 Java 平台上被编译为字节码格式，然后可以在实现这个 Java 平台的任何系统中运行。在运行时，Java 平台中的 Java 解释器对这些字节码进行解释执行，执行过程中需要的类在联接阶段被载入到运行环境中。

9）Java 是高性能的：与解释型语言相比，Java 是高性能的，随着 JIT(Just In Time)编译器技术的发展，Java 的运行速度已经越来越接近于 C++。

10）Java 语言是多线程的：在 Java 语言中，线程是一种特殊的对象，它必须由 Thread 类或其子（孙）类来创建。通常用如下两种方法来创建线程。

- 使用 Thread()的构造方法将一个实现了 Runnable 接口的对象包装成一个线程。
- 从 Thread 类派生出子类并重写 run 方法，使用该子类创建的对象即为线程。

需要注意的是 Thread 类已经实现了 Runnable 接口，所以任何一个线程均有它的 run 方法，而 run 方法中包含了线程所要运行的代码。线程的活动由一组方法来控制。Java 语言支持多个线程的同时执行，并提供多线程之间的同步机制（关键字为 synchronized）。

11）Java 语言是动态的：Java 语言能够适应动态变化的环境。Java 程序中的类能够动态地被载入到运行环境，也可以通过网络来载入所需要的类。另外，Java 中的类有一个运行时刻的表示，能够时刻检查运行的类型。

1.1.3　Java 语言的魅力

由前面讲述的 Java 特点可知，Java 是一种简单的、面向对象的、分布式的、解释型的、健壮安全的、结构中立的、可移植的、性能优异的、多线程的动态语言。自从 Sun 公司推出 Java 语言之后，全世界的目光都被这种神奇的语言所吸引。那么 Java 到底有何神奇之处呢？下面将简要介绍 Java 语言的魅力。

Java 语言最早诞生于 1991 年，起初被称为 OAK 语言，是 Sun 公司为一些消费型电子产品而设计的一个通用环境。最初的目的只是为了开发一种独立于平台的软件技术，而且在网络出现之前，OAK 是默默无闻的，当网络出现后，彻底改变了 OAK 的命运。

在 Java 出现以前，互联网上的信息内容都是一些乏味的 HTML 文档。这对于迷恋 Web 浏览的用户来说简直不可容忍。他们迫切希望能在 Web 中看到一些交互式的内容，开发人员也希

望能够在 Web 上创建一些无需考虑软硬件平台就可以执行的应用程序，当然这些程序需要有极大的安全保障为前提。对于用户的上述要求，传统的编程语言无能为力。Sun 的工程师们从 1994 年起，开始将 OAK 技术应用于 Web 上，并且开发出了 HotJava 的第一个版本。当 Sun 公司 1995 年正式以 Java 这个名字推出这种语言的时候，几乎所有的 Web 开发人员都被 Java 吸引，于是 Java 成了一颗耀眼的明星，变成了一只白天鹅，这只白天鹅越飞越高，如今，正值青春年少，它的身上焕发出耀眼的光芒，带领程序员去实现自己的梦想。

1.2　搭建开发平台

JDK 是 Java 语言的运行环境，要想使用 Java 开发应用项目，就必须先搭建 JDK。如果没有 JDK，Java 程序将无法编译和运行，因为它们中包含着许多 Java 程序。要想运行 Java 程序，必须先搭建一个平台，即先下载 JDK，然后再配置环境变量，在这之后才能够运行 Java 程序。在本节的内容中，将详细讲解搭建 Java 开发环境的基本知识。

1.2.1　下载 JDK

JDK(Java Development Kit)是 Sun Microsystems 公司（已经被 Oracle 公司收购）面向 Java 开发人员的产品。JDK 是整个 Java 的核心，包括了 Java 运行环境，Java 工具和 Java 基础的类库。JDK 是学好 Java 的第一步。

用户要运行和开发 Java 程序，必须先获得 JDK，JDK 是免费的，用户可以去官方网站获得，由于它的官方网站是英文的，可能有的初学者不知道如何下载，下载 JDK 的具体操作步骤如下。

1）打开浏览器，在地址栏中输入 http://Java.sun.com/Javase/downloads/index.jsp，然后按〈Enter〉键浏览这个网站，如图 1-1 所示。

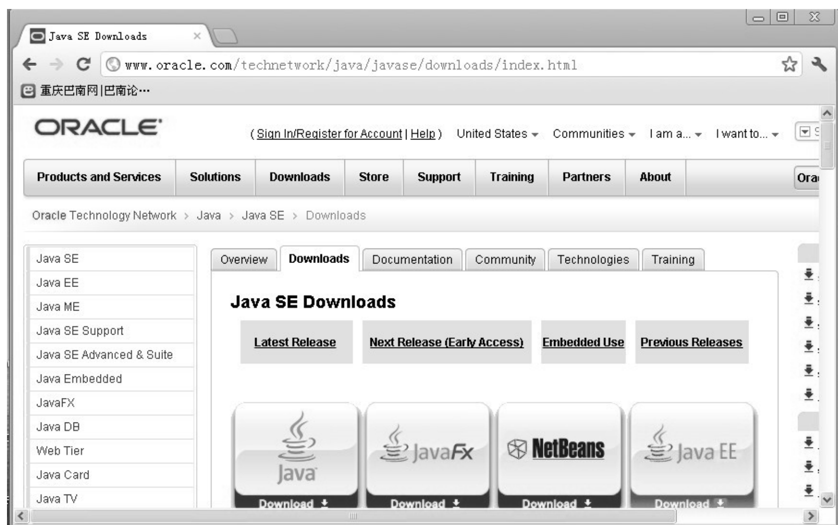


图 1-1　ORACLE 下载主页

2）在“Java SE Downloads”栏中，单击第一个图标下方的“Download”按钮，打开一

个新的页面，在“Java SE Development Kit”栏中，单击“Accept License Agreement”单选按钮，选择“Windows”版本的 JDK 如图 1-2 所示。

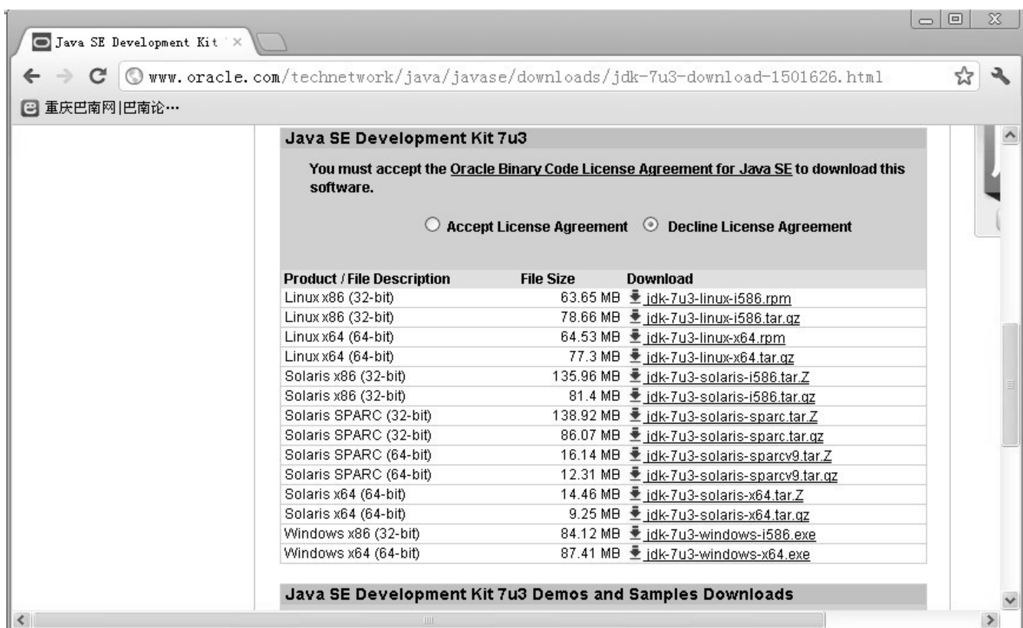


图 1-2　选择需要版本的 JDK

3）再打开的页面，即是需要下载的 JDK，用户只需双击超级链接“jdk-7u3-windows.exe”即可下载，如果安装有迅雷下载工具，下载速度将会更快，如图 1-3 所示。

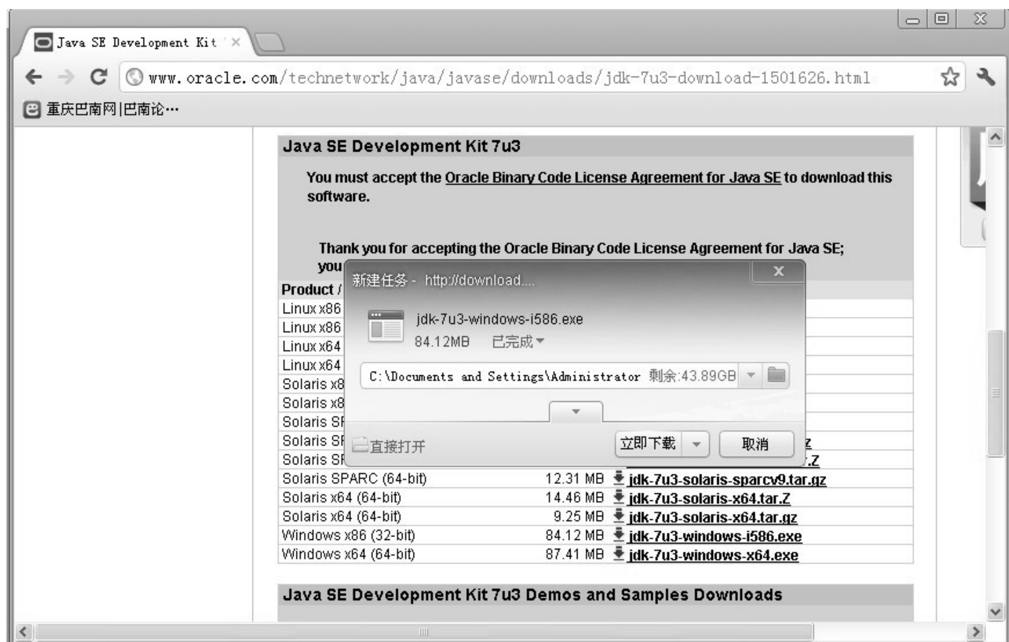


图 1-3　JDK 的下载页面

1.2.2 安装 JDK

下载 JDK 后开始进行安装，安装过程十分简单，其具体流程如下。

1）双击 JDK 安装文件，打开“自定义安装”对话框，单击“接受”按钮，如图 1−4 所示。

2）在打开的“自定义安装”对话框中选择安装的路径，这里保持默认设置，单击“下一步”按钮开始在安装路径解压缩下载的文件，如图 1−5 所示。



图 1−4 “自定义安装”对话框

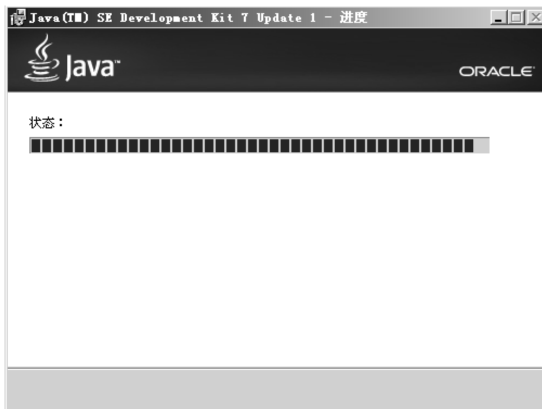


图 1−5 解压缩下载的文件

3）完成后弹出“目标文件夹”对话框，在此选择要安装的位置，如图 1−6 所示。

4）单击“下一步”正式开始安装，如图 1−7 所示。



图 1−6 选择安装 JDK 的位置



图 1−7 正式开始安装 JDK

5）完成后弹出“完成”对话框，单击“完成”按钮后完成整个安装过程，如图 1−8 所示。

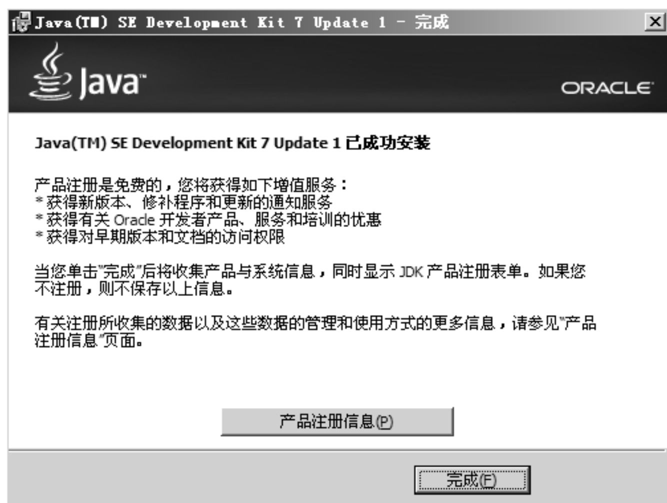


图 1−8 “完成”对话框

1.2.3　配置 JDK

安装完 JDK 后，用户需要对它进行配置，具体配置流程如下。

1）选择“计算机”图标，单击鼠标右键，在弹出的快捷菜单中选择“属性”命令，打开“系统”窗口，如图 1−9 所示。



图 1−9 “系统”窗口

2）单击“高级系统设置”打开“系统属性”对话框，单击“环境变量”按钮，如图 1−10 所示。

3）打开“环境变量”对话框，单击对话框下方“系统变量”下面的“新建”按钮，如图 1−11 所示。



图 1-10 “系统属性”对话框



图 1-11 “环境变量”对话框

4）在“新建系统变量”对话框的“变量名（N）:”文本框中输入“CLASSPATH”，然后在“变量值（V）:”文本框中设置引用包 dt.jar、tools.jar 和 rt.jar。笔者是按照默认安装进行设置的，所以此处的 CLASSPATH 的变量值是“.;C:\Program Files\Java\jdk1.6.0_13\lib\tools.jar; C:\Program Files\Java\jdk1.6.0_13\ jre\lib\rt.jar;C:\Program Files\Java\jdk1.6.0_13\lib\dt.jar;”，如图 1-12 所示。

在此初学者一定要注意，必须在输入三个文件的路径前输入“.;”。这样通过上述代码实现了对包 dt.jar、tools.jar 和 rt.jar 的引用。

5）设置完成后，单击“确定”按钮，然后单击“新建”按钮，打开“新建系统变量”对话框，输入变量名“PATH”，用户需要在“变量值（V）”文本框中输入 JDK 的 bin 路径，输入 jre 的 bin 文件夹的路径，如图 1-13 所示。

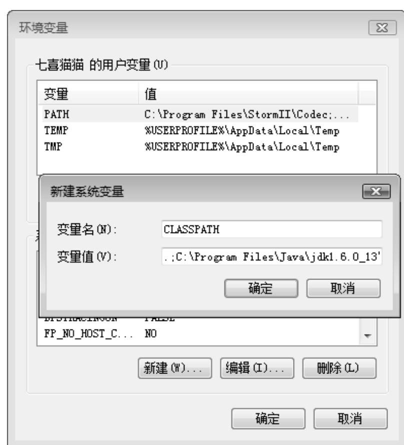


图 1-12 “新建系统变量”对话框



图 1-13 设置 PATH

提示：本例按照默认值设置的安装路径，所以它的 PATH 的变量值是“C:\Program Files\ Java\jdk1.6.0_13\bin;C:\Program Files\Java\jdk1.6.0_13\jre\bin;”。

6）通过开始菜单，选择“Windows PowerShell 1.0”程序，然后输入“javac”，将会打开如图 1-14 所示的系统信息。



图 1-14　系统信息

1.3　一个 Java 程序

安装并配置 JDK 后，就可以编写并测试 Java 程序了。在本节的内容中，将编写一个 Java 程序，让读者体验 Java 语言的魅力。

1.3.1　编写一个 Java 程序

安装并配置好了 JDK 之后，就可以开始编译和与运行代码了，下面将讲解一个具体的 Java 程序的代码，其具体操作如下。

1）打开任意一个磁盘，在“组织”下拉列表中，选择“文件夹和搜索选项”选项，如图 1-15 所示。

2）单击“查看”选项卡，在“高级设置”列表中，取消 ☑ 隐藏已知文件类型的扩展名 复选框，单击 确定 按钮，如图 1-16 所示。



图 1-15　选择“文件夹和搜索选项”选项

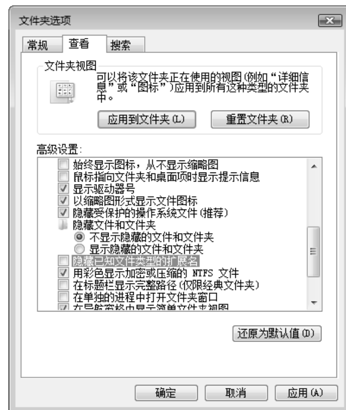


图 1-16　“文件夹选项”对话框

3）新建一个记事本，然后在记事本里，输入下面的程序：

```
class Hello{
    /**
    * 这是一个 main 方法
    */
    public static void main(String [] args){
        /* 输出此消息 */
        System.out.println("Hello，欢迎来到 Java 世界！");
    }
}
```

输入完成后，选择“文件” | “保存”命令，如图 1-17 所示。



图 1-17　输入 Java 程序

4）将这个文件命名为“Hello.java”，打开“重命名”对话框，单击“是(Y)”按钮，如图 1-18 所示。

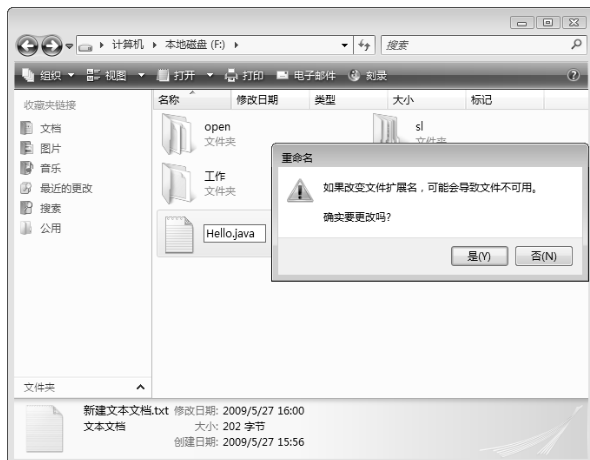


图 1-18　修改文件名和后缀名

1.3.2　Java 程序的编译与运行

当编写好一个 Java 程序后，接下来的工作就是编译并运行这段程序。编译并运行 Java

程序的方法十分简单，具体操作步骤如下。

1）单击 Windows 按钮，选择“所有程序”，然后选择“windows PowerShell”程序，如图 1-19 所示。



图 1-19　选择程序

2）进入 Java 文件夹下，例如 F：盘，然后输入 javac 命令，按〈Enter〉键编译源文件，如图 1-20 所示。

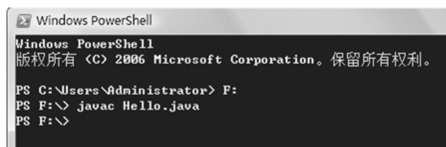


图 1-20　编译 Java 源文件

3）打开存放源文件的文件夹，仔细观察，会发现有一个后缀名为.class 的文件夹，如图 1-21 所示。

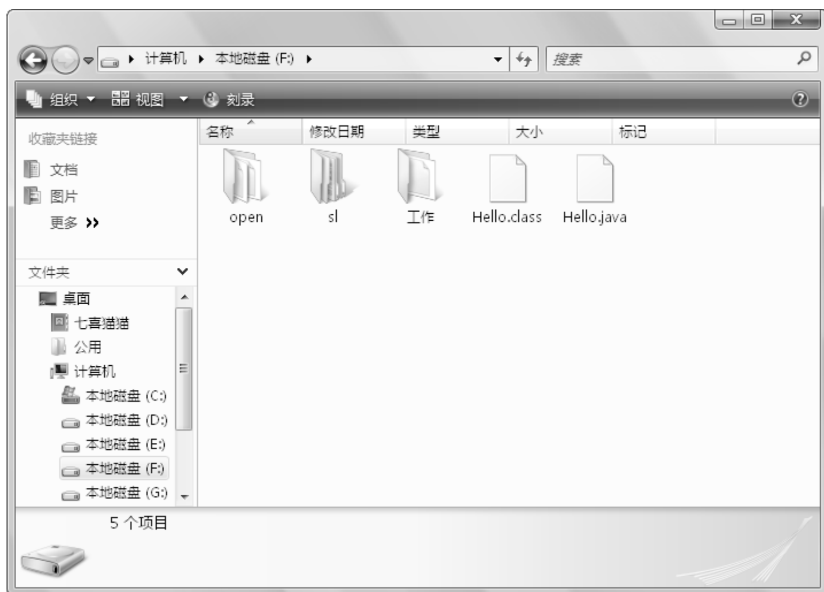


图 1-21　编译后的文件夹

4）当编译后输入 Java 命令，然后按〈Enter〉键，如图 1-22 所示。

提示：如果源文件不是在磁盘的根目录下，用户可以通过两种方式进入这个目录，可以用 cd 命令打开文件夹，也可以复制“Windows PowerShell 1.0”快捷方式。

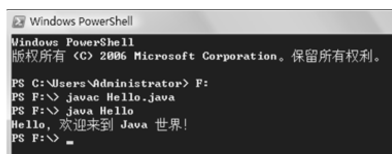


图 1-22　执行后的结果

第一种方式：使用命令打开非根目录下文件夹。

假如要打开 F 盘下的“open”文件夹，需要先在 cd 窗口输入盘号“F”按〈Enter〉键，然后输入“cd open”，这样就成功进入 F 盘的子文件夹“open”，如图 1-23 所示。

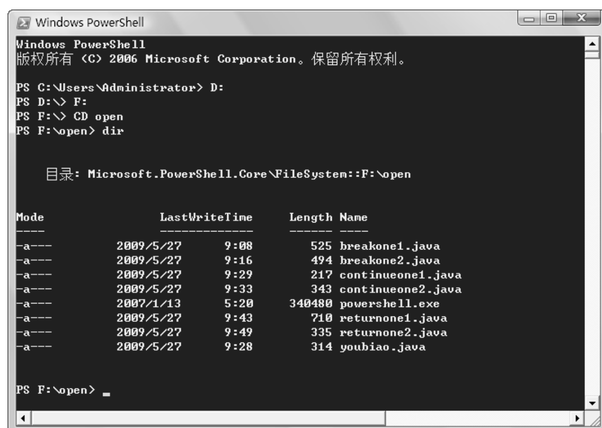


图 1-23　输入命令

第二种方式：复制快捷方式。

在 C 盘输入“Windows PowerShell”关键字进行搜索，如图 1-24 所示。然后将搜索到的快捷方式复制到需要打开的文件夹下，此时只需要打开这个快捷方式即可进入要打开的那个文件夹中。



图 1-24　搜索出的快捷方式

1.4　疑难问题解析

在本章的内容中，详细讲解了 Java 语言的最基本知识，体会了 Java 在编程语言中的重要地位。本节将对本章中比较难以理解的问题进行讲解。

读者疑问：开发其他程序有开发工具，开发 Java 有开发工具吗？

解答：Java 是一门优秀的语言，它当然有开发工具，其中的 Eclipse 非常强大，它在开发 Java 程序的各个方面，表现得十分出色，本章的目的是让读者理解 Java 的源程序，所以没有使用 Eclipse 实现开发，在本书后面的内容中，将讲解使用开发工具的知识，Eclipse 的界面如图 1-25 所示。



图 1-25　Eclipse 开发工具

读者疑问：如果要编译多个 Java 源文件，有没有更简单的方法？一个一个的编译文件有点麻烦。

解答：方法是有的，只需要输入 javac *.java,就可以编译全部的 Java 文件，如图 1-26 所示。

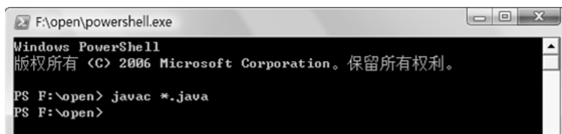


图 1-26　编译 Java 文件



职场点拨——谈 Java 的地位

前面读者知道了 Java 有众多的开发工具这一优势，其实 Java 的优势何止如此。先看国

外某编程社区做的 2009 年和 2010 年编程语言使用率统计数据，具体如表 1-1 所示。

表 1-1　2009 年和 2010 年编程语言使用率统计表

2010 年排名	2009 年排名	语　　言	2010 年占有率(%)	和 2009 年相比(%)
1	2	C	18.058	+2.59
2	1	Java	18.051	−1.29
3	3	C++	9.707	−1.03
4	4	PHP	9.662	−0.23
5	5	Visual Basic	6.392	−2.70
6	6	C#	4.435	+0.38

由表 1-1 的统计数据可以看出，Java 语言一直占据当前使用率排名的前两位。

（1）看 Java 的主要应用领域

Java 的功能比较强大，在服务器领域、移动设备、桌面应用和 Web 领域都占据了重要的地位。

- 服务器领域：Java 在服务器编程方面很强悍，拥有很多其他语言所没有的优势。
- 移动设备：Java 在手机领域的应用比较广泛，手机 Java 游戏随处可见，当前异常火爆的 Android 手机也支持 Java。
- 桌面应用：Java 和 C++、.NET 一样重要，影响着桌面程序的发展。
- Web 领域：Java Web 有着巨大的优势，无论是开发工具还是开发框架都是开源的，并且安全性更强。

（2）就业前景

全球有 25 亿台 Java 器件运行着 Java，450 多万 Java 开发者活跃在地球的每个角落，数以千万计的 Web 用户每次上网都亲历 Java 的威力。今天，Java 运行在 7.08 亿部手机、10 亿个智能卡和 7 亿台 PC 上，并为 28 款可兼容的应用服务器提供了功能强大的平台。这么多应用，彻底改变了用户的生活。越来越多的企业，因为使用了 Java 而提高了生产效率。在中国，越来越多的用户，因为 Java 而降低了成本，享受了生活。

作为唯一在互联网上开发的语言，Java 平台以其移动性、安全性和开放性受到追捧。据 IDC 预计，自 2010 年起的其后 5 年内，采用 Java 的 IT 产品的价值将翻番，在 2015 年将达到 45.53 亿美元，年增长率为 14.9%。截止到 2010 年 5 月，Java 注册开发商超过 1300 万人，对 JRE(Java 运行环境)的下载达 9200 万次。詹姆斯·戈士林博士预计在 3～5 年内 Java 技术开发商将发展到 2000 万，无线 Java 也在迅速攀升。

目前，我国对软件人才的需求已达 40 万，并且以每年 20%左右的速度增长。在未来 5 年内，合格软件人才的需求将远大于供给。2010 年，我国软件人才的缺口已达 42.5 万，其中尤以 Java 人才最为缺乏。

根据 IDC 的统计数字，在所有软件开发类人才的需求中，对 Java 工程师的需求达到全部需求量的 60%~70%。这也就不难解释在智联招聘和 51Job 等这些主流招聘网站，随处可见的 Java 工程师的招聘广告了。

第 2 章　Java 语法

Java 语法是 Java Web 开发应用的根本，只有遵循 Java 语言的语法格式，才能构建一个完整的、具备一定功能的 Java 程序。所以在本章的内容中，将向读者详细讲解 Java 语法的基本知识。通过本章能学到如下知识。

- 标识符和注释。
- 变量和常量。
- 数据类型。
- 数据之间的转换。
- 运算符与表达式。
- 职场点拨——扎实的基础很重要。

背景介绍

2008 年 X 月 X 日，天气阴

今天我编写了一段 Java 程序，但是在调试时总是提示编译错误。只好向老师请教，老师看后仿佛很好笑地说道："你的变量定义错了，这样的错误你都犯，看来上课没注意听讲啊！"



一问一答

小菜："今天真丢人，一个不经意的粗心，浪费了半天时间才调试成功！"

Wisdom："呵呵，程序开发是一个细心的工作，哪怕是错一个字符，程序都不会成功运行！"

小菜："嗯，言归正传，本章的语法知识很重要吗？"

Wisdom："当然了，这是 Java 开发的基础，语法是一种编程语言最基础的知识。在本章的内容中，将系统介绍 Java 的语法，包括变量、数据类型和运算符等。"

2.1　认识 Java 代码

在编写 Java Web 之前，肯定要认识 Java 代码，Java 基础语法十分简单，下面以一段简单的代码为例，作为本章的引子，其代码（光盘：源代码/第 2 章/kai.java）如下：

```
public class Kai
```

```
{
 int a1=127;
 int s1=a1*a1;
 public static void main(String args[])
{

          //长方形面积
         double a2=7.1,b2=4.3;
         double s2=a2*b2;
         System.out.println("长方形的面积为"+s2);
         //三角形面积
         double a3=5.2,b3=6.7;
         double s3=a3*b3/2;
         System.out.println("三角形的面积为"+s3);
 }
}
```

将代码进行编译，然后运行，得到如图 2-1 所示的结果。

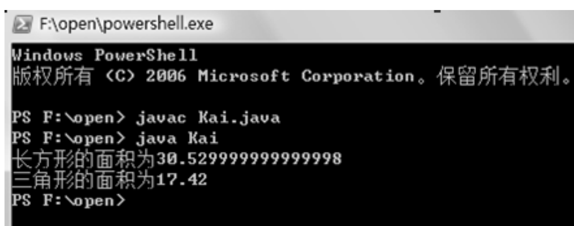


图 2-1　编译和执行程序

2.2　标识符和注释

在 Java 语言中，标识符和注释是必不可少的语法内容。其中标识符是以字母、下画线或美元符号开头，由字母、数字、下画线或美元符组成的字符串；而注释是用来解释和标注 Java 源程序的代码。在本节的内容中，将一一讲解标识符和注释的基本知识。

2.2.1　神奇的标识符

标识符是赋给类、方法或变量的名称，在 Java 语言中，通常用标识符来识别类名、变量名、方法名、类型名、数组名和文件名。

在 Java 中规定，标识符由大小写字母、数字、下画线“_”、美元符号“$”组成，但不能以数字开头，标识符没有最大长度限制。例如下列都是合法的标识符。

Chongqin_$　D3Tf　T_w_o　$67.55

可以通过如下方法判断标识符是否不合法。

- 标识符不能以数字开头，如 7788。
- 标识符中不能出现规定以外的字符，如 You’are、deng@qq.com。

- 标识符是严格区分大小写的，在 Java 中，no 和 No 是完全不同的，还需要注意的是虽然使用$符号在语法上是被允许的，但在编码规范中规定尽量不要使用它，因为它容易混淆。

2.2.2　注释的力量

在 Java 程序中，注释语句只起了注释的作用。虽然 Java 中的注释在程序中并不执行，对程序本身不产生任何影响，但是程序中的注释是必不可少的，如在一个多人协作的大项目中，通过注释可以让团队成员更加容易理解每段程序的作用。

实例 1：使用注释

在 Java 中，注释语句分为单行注释与多行注释两种。在下面的代码中，笔者为程序添加了一些注释，既有单行注释也有多行注释。读者通过这些注释语句可以明白这段代码的含义。

```
public class JieShi
{
    public static void main(String[] args)
      {
         flower1 f1=new flower1();
         flower1 f2=new flower1();
         f1.setcolor("红色");
         f1.setname("牡丹");
         f1.setcountry("中国");
         f1.setcity("沈阳");
         f2.setcolor("黄色");
         f2.setname("玫瑰");
         f2.setcountry("美国");
         f2.setcity("拉斯维加斯");
         System.out.println(f1.tostring());
         System.out.println(f2.tostring());
      }
   }
   ///这是一个花的接口
   ///其中的方法动没有被实现
   interface flower
   {
      void setcolor(String color1);
      void setname(String name);
      String  getcolor();
      String getname();
   }
   //这是一个花的类，实现了上面的那个接口
   //通过设置器来设置对象的参数
   //通过访问器来获得对象的参数
   //通过 tostring 方法来让对象以字符串的形式输出
```

```
class flowers implements flower
{
   private String color1;
   private String name;
   public void setcolor(String color1)
   {
      this.color1=color1;
   }
   public void setname(String name)
   {
      this.name=name;
   }
   public String getcolor()
   {
      return color1;
   }
   public String getname()
   {
      return name;
   }
   public String tostring()
   {
     String information="花的名称是："+name+";"+" "+"花的颜色是："+color1;
      return information;
   }
}
/*这是一个花的类，继承了上面的那个类
*通过设置器来设置对象的参数
*通过访问器来获得对象的参数
*通过 tostring 方法来让对象以字符串的形式输出**/
class flower1 extends flowers
{
   private String country;
   private String city;
   public void setcountry(String country)
   {
      this.country=country;
   }
   public void setcity(String city)
   {
      this.city=city;
   }
   public String getcountry()
   {
      return country;
   }
```

```
            public String getcity()
            {
               return city;
            }
            public String tostring()
            {
               String information=super.tostring()+"   "+"这种花出自的国家：
"+country+";"+"  "+"出自的城市："+city;
               return information;
            }
       }
```

将上述代码进行编译并运行，得到如图 2-2 所示的注释结果。

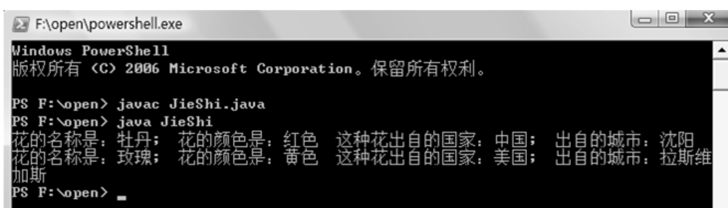


图 2-2　注释结果



在 Java 程序中，注释语句并不执行，用户可以通过两种方式对程序进行注释。单行注释语句只能在一行内生效，其格式如下:

```
       //注释内容
```

多行注释语句可以注释多行，其注释格式如下:

```
       /*注释内容
       *注释内容
       **/
```

2.3　变量和常量

在 Java Web 程序中，因为有太多的 Java 程序，所以也会有太多的变量和常量，所以认识 Java 中的变量和常量必不可少。在下面的内容中，将对变量和常量的基本知识进行详细讲解。

2.3.1　变量

无论在什么样的程序设计中，都会用到变量。在程序运行过程中，如果内存空间内的值是变化的，那么这个内存空间就称为变量。为了操作方便，给这个空间取名为变量名。因为内存空间内的值是变量值，所以即使申请了内存空间，变量也不一定有值。要想变量有值，

就必须要放入值，但申请的变量，不同的数据类型，它们将会有一个默认值，如 int 的数据变量的默认值是“0”，char 的数据变量的默认值是 null，byte 的数据变量的默认是“0”。

在程序运行中，变量的基本格式与常量有所不同。变量的基本格式如下：

```
typeSpecifier varName=value;
```

typeSpecifier 为 Java 中合法的数据类型，这和常量是一样的，varName 为变量名，变量和常量的最大区别是 value 的值是可有可无的，而且还可以对其进行动态初始化。

变量分为局部变量和全局变量两种，全局变量中也称做成员变量，该变量被定义在一个类中，在所有的方法和函数之外，局部变量在一个方法或者一个函数中。

实例 2：使用变量计算正方形、三角形和长方形的面积

如果在局部范围内访问全局变量，需要使用一个关键字，下面通过一个代码进行讲解，其代码（光盘：源代码/第 2 章/Liang.java）如下：

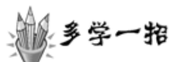
```
public class Liang
{
   public static void main(String args[])
   {
         //正方形面积
         double a1=9.2;
         double s1=a1*a1;
         System.out.println("正方形的面积为"+s1);
         //长方形面积
         double a2=6.1,b2=8.3;
         double s2=a2*b2;
         System.out.println("长方形的面积为"+s2);
         //三角形面积
         double a3=7.0,b3=5.0;
         double s3=a3*b3/2;
         System.out.println("三角形的面积为"+s3);
   }
}
```

将代码进行编译并运行，得到如图 2-3 所示的结果。

```
F:\open\powershell.exe
Windows PowerShell
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

PS F:\open> javac Liang.java
PS F:\open> java Liang
正方形的面积为84.63999999999999
长方形的面积为50.63
三角形的面积为17.5
PS F:\open> _
```

图 2-3　使用变量计算的结果



上述实例中的变量是一个局部变量，只在一个方法块或者一个函数内起作用，如果超过这个范围，它将没有任何作用。而在下面的代码中，虽然全局变量可以在整个程序中产生作用，但是在局部可以随时更改这个变量的值，其代码（光盘：源代码/第 2 章/var.java）如下：

```
public class var
{
   byte x;
   short y;
   int z;
   int z1;
   long a;
   float b;
   double c;
   char d;
   boolean e;
   public static void main(String[] args)
   {
        int z1=111;
      System.out.println(" 打印数据 z="+z1);
      var m=new var();
      System.out.println(" 打印数据 x="+m.x);
      System.out.println(" 打印数据 y="+m.y);
      System.out.println(" 打印数据 z="+m.z);
      System.out.println(" 打印数据 a="+m.a);
      System.out.println(" 打印数据 b="+m.b);
      System.out.println(" 打印数据 c="+m.c);
      System.out.println(" 打印数据 d="+m.d);
      System.out.println(" 打印数据 e="+m.e);
   }
}
```

因为在上述代码中，除 z 之外使用了各种类型的默认值，所以在控制台中的输出结果是 0 或空值等。将上述代码进行编译并运行后得到如图 2-4 所示的结果。

```
F:\open\powershell.exe
Windows PowerShell
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

PS F:\open> javac var.java
PS F:\open> java var
 打印数据z=111
 打印数据x=0
 打印数据y=0
 打印数据z=0
 打印数据a=0
 打印数据b=0.0
 打印数据c=0.0
 打印数据d=
 打印数据e=false
PS F:\open>
```

图 2-4　全局变量

2.3.2 常量

理解了变量后，常量就容易理解了。常量是指不会变化的量，在 Java 中的常量分为整型常量、浮点常量、字符常量、布尔常量和字符串常量，常量的基本格式如下：

```
final double PI=value;
```

参数 value 是该数据合法的值。

2.4 数据类型

在 Java Web 中，不管是 Java 程序里，还是 Web 程序里，数据类型都和 Java 程序相同，可以将数据类型分为简单数据类型和复合数据类型，简单类型是 Java 中最基本的数据类型，包括整数类型、浮点类型、字符类型和布尔类型等。复合数据类型由简单数据类型组成，是用户根据自己的需要定义并实现其运算的类型，如类、接口、数组等，在本书后面的内容中将为读者进行一一讲解。

2.4.1 了解简单数据类型

在学习数据类型之前，读者一定要了解简单数据类型，它有哪些，各自的取值范围又是什么，如表 2-1 所示。

表 2-1 数据类型

数 据 类 型	所 占 位 数	值 范 围
byte（字符类型）	8 位	−128~127
short（短整型）	16 位	−32768~32767
int（整型）	32 位	−2147483648~2147483647
long（长整型）	64 位	−9 223 372 036 854 775 808 到 9 223 372 036 854 775 80
float（单精度浮点型）	32 位	3.4e+/−38
double（双精度浮点型）	32 位	1.79769313486231570E+308~~1.79769313486231570E+308
char（字符型）	64 位	0~65535
boolean（布尔型）	1 位	true 或 false

2.4.2 byte 型

在 Java 中提供了一个名为 byte 的数据类型，这是一种基本类型。Java byte 是作为最小的数字来处理的，因此它的值域被定义为−128~127，也就是 signed byte。

2.4.3 整型

整型是最简单的数据类型，也是常用的数据类型，它是有符号的 32 位整数数据类型。整型 int 常常用在数组、控制语句等多个地方，Java 系统会把 byte 和 short 自动提升为整型 int。

2.4.4　浮点数据类型

浮点类型是相对整型的数据类型，用来表示 Java 中的浮点数，浮点类型数据表示有小数部分的数字，总共由两种类型组成：单精度浮点型（float）和双精度浮点型（double），下面对其进行讲解。

1．单精度浮点型

单精度浮点型是专指占用 32 位存储空间的单精度数据类型，在编程的过程中，当需要小数部分且对精度要求不高时，一般使用单精度浮点型，这种数据类型很少用，本书不作详细讲解。

2．双精度浮点型

双精度浮点型在数据计算中占有很大的比重，双精度浮点型能够保证数值的准确型，例如可以通过下面的代码计算圆的面积，代码（光盘：源代码/第 2 章/yuan.java）如下：

```
public class Yuan{
   public static void main(String args[])
   {
         double R=45.24;
         final double PI=3.1415926;
         double area=PI*R*R;
         System.out.println("圆的面积是：S="+area);
   }}
```

提示：在上面的程序中，出现了两个符号“*”、“=”，它们是运算符，第一个符号相当于数学中的乘号，第二个是赋值符号，并不是数学中的等号，这些知识点在下面的章节中会讲解。

将上述代码进行编译并运行，得到如图 2-5 所示的结果。



图 2-5　双精度浮点型

2.4.5　字符型

在程序中，存储字符的数据类型是字符数据类型，用字母 char 表示，下面请看一段代码，等读者看完代码后，再进行详细讲解，其代码（光盘：源代码/第 2 章/zifu.java）如下：

```
public class Zifu {
   public static void main(String args[]){
```

```
            char a='\u0011';
            char b='\u0894';
            char c='\uffff';
            System.out.println(a);
            System.out.println(b);
            System.out.println(c);
        }}
```

将代码进行编译并运行，得到如图 2-6 所示的结果。

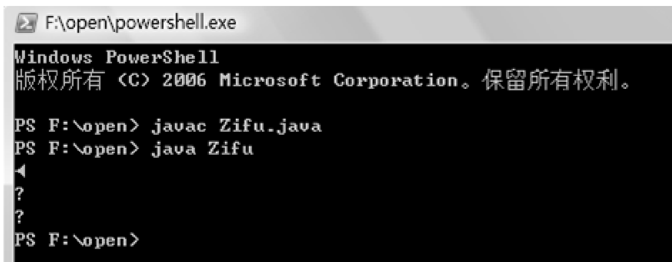


图 2-6　字符类型

通常用 Unicode 码来表示字符，所以就有了上述结果。Unicode 所定义的国际化字符集能表示常用的所有字符集，如拉丁文、希腊语等几十种语言，这些大部分是看不懂的，用户不需要掌握。读者请注意，在执行的结果处有一个问号，它有可能是真的问号，有可能是不能显示的符号。但是为了正常地输出这些符号，Java 提供了转义字符，以“\”开头，十六进制计数法用“\”和“U”字开头，后面跟着十六进制数字，常用的转义字符如表 2-2 所示。

表 2-2　转义字符

转 义 字 符	描　述
\ddd	八进制字符（ddd）
\uxxxx	十六进制 Unicode 字符（xxx）
\’	单引号字符
\”	双引号字符
\\	反斜杠
\r	回车
\n	换行
\f	走纸换页
\t	横向跳格
\b	退格

实例 3：使用转义字符

熟悉了表 2-2 中的转义字符后，下面通过一个实例讲解转义字符，其代码（光盘：源代码/第 2 章/zhuanyione.java）如下：

```
public class zhuanyione
{
   public static void main(String args[])
   {
         System.out.println("maomao\\taotao");
   }
}
```

将代码进行编译，然后运行，得到如图 2-7 所示的结果。

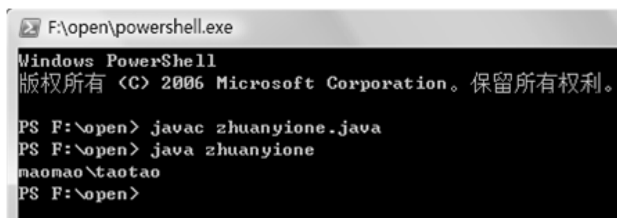


图 2-7　转义字符


多学一招

在实例中主要讲解了如何在程序中输出一些特殊字符，学过这个方法后，建议读者要记住表 2-2 所示的转义字符的使用方法。例如在下面的代码中演示了其他几种转义字符的用法，具体代码（光盘：源代码/第 2 章/zhuanyitwo.java）如下：

```
public class zhuanyitwo
{
   public static void main(String args[])
   {
         System.out.println("beijing\\shanghai");
         System.out.println("beijing\'shanghai");
         System.out.println("beijing\"shanghai");
         System.out.println("beijing\rshanghai");
   }
}
```

将代码进行编译，然后运行，得到如图 2-8 所示的结果。

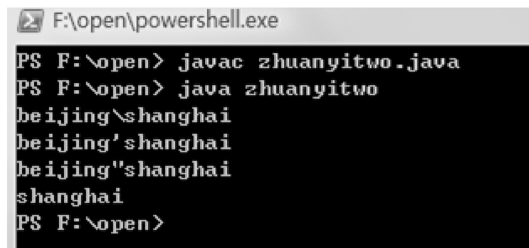


图 2-8　输出一些特殊的字符

2.4.6 逻辑型

逻辑型又称做布尔类型，是 Java 数据中用来判别正确与错误的一种数据类型。逻辑型只有两个取值，分别是 true 和 false。

实例 4：使用逻辑型字符

下面通过一个实例来讲解逻辑型的用法。本实例的实现代码（光盘：源代码/第 2 章/DuiCuo.java）如下：

```
public class DuiCuo
{
   public static void main(String args[])
   {
         int dui=6+3;
         boolean bn=(dui==9);
         System.out.println(bn);
   }
}
```

将代码进行编译，然后运行，得到如图 2-9 所示的结果。

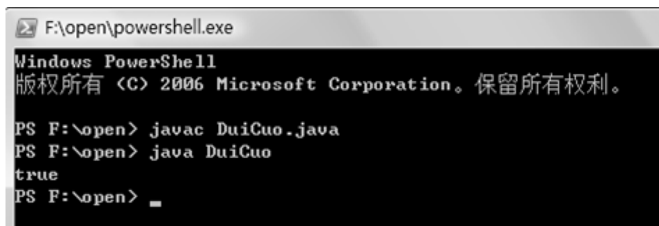


图 2-9　逻辑型


多学一招

逻辑型实际上就是一个判断的语句，只要条件符合要求，它将会给出答案 true，如果条件不符合要求，它将给出答案 false。为了加深读者对逻辑型的理解，再看下面的一段代码（光盘：源代码/第 2 章/DuiCuo1.java）：

```
public class DuiCuo1
{
   public static void main(String args[])
   {
         int tan=1;
         boolean bn=(tan>9);
         System.out.println(bn);
   }
}
```

将上述代码进行编译并运行，得到如图 2-10 所示的结果。

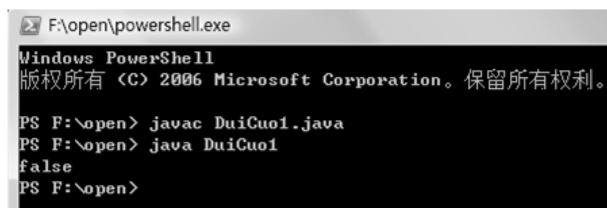


图 2-10　布尔类型

2.5　数据之间的转换

在开发 Java Web 的 Java 程序代码里，通常会遇到两种不同数据类型之间的计算，如 char a 和 int b 相加，这时候就需要使用类型转换。在 Java 中，有时两种数据类型可以兼容，例如一个 byte d 数据可以赋值给一个 int h 数据，int h 数据再将它传给 long j 数据，在这个过程中，Java 就能实现自动数据类型的转换，否则 Java 就得进行强制转换，下面对自动数据类型的转换与强制数据类型的转换进行详细讲解。

2.5.1　自动类型转换

在编写程序时，数据类型会自动转换，此时只需要满足两个条件即可，即两个数据类型兼容和目标数据类型要大于原数据类型，整数和浮点数、数字类型和布尔类型、字符型和逻辑型是不兼容的。为了加深读者对上述内容的理解，下面通过一个实例进行讲解，让读者体会数据之间的兼容。

实例 5：实现自动类型转换

通过本实例，读者将会看到一些数据相加后的执行结果，其代码（光盘：源代码/第 2 章/Zidong.java）如下：

```
public class Zidong
{
   public static void main(String args[])
   {
         byte a=17;
         short b=12;
         int c=6;
         float d=3.0f;
         double e=9.0;
         System.out.println(a+b);
         System.out.println(b+c);
         System.out.println(d+e);
         System.out.println(a+b+c+d+e);
   }
}
```

将上述代码进行编译并运行，得到如图 2-11 所示的结果。



图 2-11　自动类型的转换



多学一招

实例 5 中的数据是可以兼容的，但如果是不兼容的数据，那么系统会给出什么样的提示呢？为了找到答案，请读者看下面的一段代码（光盘：源代码/第 2 章/zidong1.java）。

```
public class Zidong1
 {
   public static void main(String args[])
   {
         int a=17;
         boolean b=true;
         char c='e';
         System.out.println(a+b);
         System.out.println(b+c);
   }
}
```

将上述代码进行编译并运行，得到如图 2-12 所示的结果。



图 2-12　不兼容的数据

根据错误提示，不兼容的数据类型也可以实现混合运算，例如，整型、浮点型和字符型等。在 Java 中运行不同类型的计算时，需要先将不同类型的数据类型转换为同一类型，然后再进行计算，如表 2-3 所示，左边的数据类型可以向右边的数据类型转换。

表 2-3　不同类型转换规则

数据类型 1	数据类型 2	数据类型 3
byte short	int	int
byte short int	long	long
byte short int long	float	float
byte short int long float	double	double
char	int	int

2.5.2　强制类型转换

经过前面的学习，了解了范围小的类型可以向范围大的类型进行自动转换。也许有读者要问，怎样将范围大的 int 型值赋给一个范围小的 byte 类型，这样会造成数据丢失，这时候应该怎么处理呢？在 Java 中，可以用强制转换将范围大的类型转换为范围小的类型，强制转换类型的格式如下：

```
（target-type）value
```

实例 6：把 int 型强制转换成 byte 型

在本实例中，用 Target-type 指定了要转换的类型，把 int 型强制转换成了 byte 型，其代码（光盘：源代码/第 2 章/qiang.java）如下：

```
public class qiang
{
   public static void main(String args[])
   {
         int a=586;
         byte b=(byte)a;
         System.out.println(a);
         System.out.println(b);
   }
}
```

将上述代码编译并运行，得到如图 2-13 所示的结果。

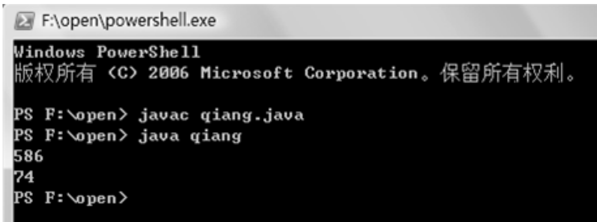
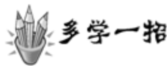


图 2-13　强制数据类型的相加



看到实例 6 的结果，读者可能会很惊讶，为什么 b 的值是 74 呢？这是因为 b 的值超过

了 byte 的取值范围，它的值将会因为 byte 型的值域而减小，它的值是这个数除以 byte 的取值个数（−128~127 共 256），得到的余数，当浮点型赋给整数类型时，它的小数部分将会被舍去。如果浮点型舍去小数还不在其取值范围内，则它将采用上面的整除取余数的方法，其代码（光盘：源代码/第 2 章/qiang1.java）如下：

```
public class qiang1
 {
   public static void main(String args[])
   {
         double a=7.234;
         double b=7876.294;
         byte c=(byte)a;
         System.out.println(c);
         byte d=(byte)b;
         System.out.println(d);
   }
}
```

将上述代码编译并运行，得到如图 2−14 所示的结果。

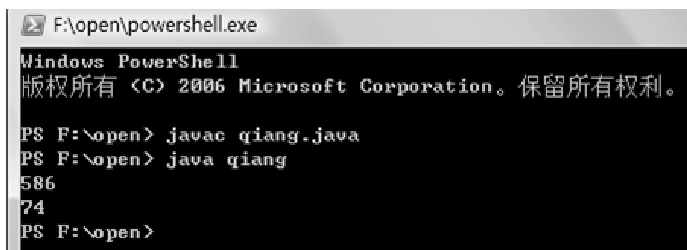


图 2−14　整数取余的转换

2.6　运算符与表达式

在 Java Web 里，使用 Java 运算符是十分常见的。Java 中运算符的种类有很多，如算术运算符、位运算符、关系运算符、逻辑运算符等。将操作数和运算符连接起来，就形成了表达式。运算符和表达式都是程序的基础，下面将对运算符和表达式的知识一一讲解。

2.6.1　运算符

Java 中，运算符分为算术运算符、位运算符、关系运算符、逻辑运算符和其他运算符，下面将对这些运算符一一讲解。

1．算术运算符

算术运算符号用于算术表达式，其功能和用法与数学中的运算符一样，它由基本算术运算符、取余运算符等组成，如表 2−4 所示列出了 Java 中的算术运算符。

表 2-4　算术运算符

类　　型	运　算　符	说　　明
基本运算符	+ - * /	加 减 乘 除
取余运算符	%	取模
递增或递减	++ --	递增 递减

基本运算符和取余运算符十分简单，就和数学中的用法一样。但是递增和递减对初学者很难理解，下面将通过一段演示代码进行讲解，其代码（光盘：源代码/第 2 章/yun1.java）如下：

```
public class yun1 {
   public static void main(String args[])
   {
         int a=7199;
         int b=2009;
         System.out.println(a++);
         System.out.println(a);
         System.out.println(++a);
         System.out.println(b--);
         System.out.println(b);
         System.out.println(--b);
   }
}
```

将上述代码编译并运行，得到如图 2-15 所示的结果。

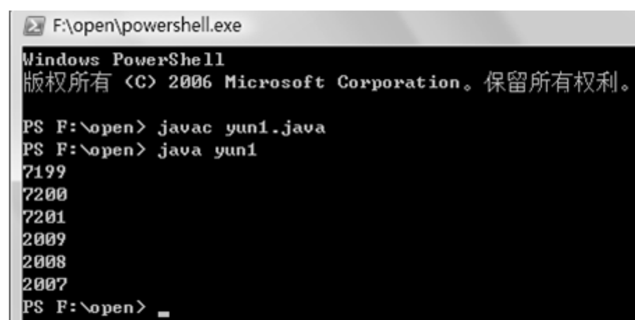


图 2-15　算术运算符

从图 2-14 的执行结果可以看出，a++是先执行程序才加 1，++a 是先加 1 再执行程序，a--是先执行再减 1，--a 是减 1，再执行程序。“system.out.println(a++);”它应是先执行再加 1，所以它输出的值应该是初始值 7199，“system.out.println(a);”因为前一句话将其加 1，所以结果有所变化，结果为 7200，“system.out.println(++a)”是先加 1，再执行程序，执行的结果是 7201，后面的程序以同样的道理得到结果，初学者应该多多理解。

2．关系运算符

关系运算符用来确定一个操作数和另外一个操作数之间的关系，在表 2-5 中列出了 Java 中的关系运算符，通过这些关系运算符，将会产生一个结果，这个结果是一个布尔值，即 true 和 false，在 Java 中任何类型的数据都可以用“==”进行比较是不是相等，用“!=”比较不等。只有数字才能比较大小，关系运算的结果可以直接赋予布尔变量。

表 2-5　关系运算符

类　　型	说　　明
==	等于
!=	不等于
>	大于
<	小于
>=	大于等于
<=	小于等于

3．逻辑运算符

布尔逻辑运算符是最常见的逻辑运算符，用于对 Boolean 型操作数进行布尔逻辑运算，Java 中的布尔逻辑运算符信息如表 2-6 所示。

表 2-6　布尔逻辑运算符

类　　型	说　　明
&&	与(AND)
‖	或（OR）
∧	异或（XOR）
|	简化或（Short-circuit OR）
&	简化或（Short-circuit AND）
!	非（NOT）

下面通过一段代码来讲解布尔逻辑运算符的基本用法，其代码（光盘：源代码/第 2 章/luoji.java）如下：

```
public class luoji
{
   public static void main(String[] args)
   {
         int d=10;
         int k=12;
         int j=111;
         System.out.println(d>k||k<j);
         System.out.println(d>k&&k<j);
         System.out.println(!(d>k));
   }
}
```

将上述代码编译并运行，得到如图 2-16 所示的结果。

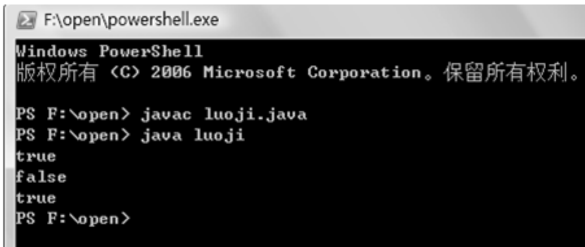


图 2-16　逻辑运算符

4. 位运算符

位运算符是用来对二进制数进行操作的。位运算符一般分为两种，分别是位逻辑运算符和移位运算符，Java 中的位运算符信息如表 2-7 所示。

表 2-7　位运算符

位逻辑运算符	说　明
~	按位取反运算
&	按位与运算
|	按位或运算
^	按位异或运算
>>	右移
>>>	右移并用 0 填充
<<	左移

如表 2-8 所示是操作数 A 和操作数 B 按位逻辑运算的结果。

表 2-8　位逻辑运算结果

操作数 A	操作数 B	A|B	A&B	A^B	~A
0	0	0	0	0	1
0	1	1	0	1	1
1	0	1	0	1	0
1	1	1	1	0	0

通过如下一段程序进行讲解。

```
public class Wei
{
  public static void main(String args[])
  {
      int a=9;
      System.out.println(a);
      a=a>>2;
      System.out.println(a);
```

```
    }
}
```

将上述代码进行编译并运行，得到如图 2-17 所示的结果。

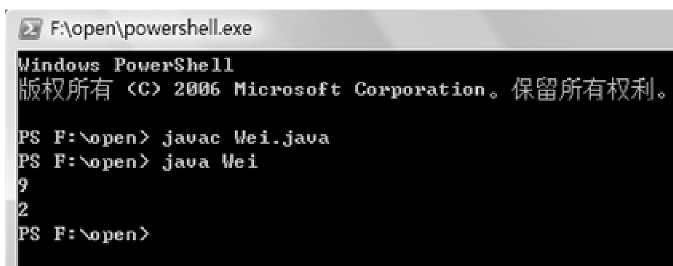


图 2-17　位运算符

5．赋值运算符和条件运算符

赋值运算符是一种很简单的运算符，在本书前面的内容中已经接触过。条件运算符是一种特殊的运算符，与本书前面所讲解的运算符有很大不同。在 Java 程序中，条件运算符的目的是决定把哪儿的值赋给前面的变量，其基本结构如下：

```
变量=（布尔表达式）？为 true 时所赋予的值：为 false 时所赋予的值；
```

实例 7：使用条件运算符判断是否及格

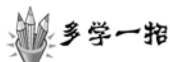
下面将通过一个具体实例，让读者加深对上面条件运算符的理解，其代码（光盘：源代码/第 2 章/Tiao.java）如下：

```
public class Tiao
 {
   public static void main(String args[])
   {
         double chengji=60;
         String jieguo=(chengji>=60)?"我及格了":"没有及格";
         System.out.println(jieguo);
   }
}
```

将上述代码进行编译并运行，得到如图 2-18 所示的结果。



图 2-18　条件运算符



在实例 7 中，可根据条件运算符的结构语句判断结果，最终结果是“我及格了”。如果条件为真，应该取第一个值，如果条件为假，应该取第二个值。

下面再写一段代码，用户先判断出一个结果，然后进行编译并执行，查看运行结果是否与自己判断的相同，其源代码如下:

```
public class Tiao1 {
   public static void main(String args[])
   {
         double chengji=70;
         String Tiao=(chengji>=90)?"已经很优秀了":"不是很优秀，我还需要努力！";
         System.out.println(Tiao);
   }
}
```

将上述代码进行编译并运行，得到如图 2-19 所示的结果。



图 2-19　不满足条件执行的结果

2.6.2　表达式

在本书前面的章节中，已经多多少少接触了一些表达式。但是对表达式还十分迷糊，并没有深刻理解表达式。在接下来的内容中，将详细讲解 Java 中表达式的基本知识。

在程序代码内部，每个表达式都有自己对应的数据类型，例如，int n = 10，int m = 10;，这些都是带着数据类型的表达式，在实际的程序代码中，大部分的表达式不能单独成为代码中的一行，否则程序会提示语法错误，例如：

```
int a = 10;
int b = 20;
a + b;//不能单独成行
```

在表达式中，能够单独成行的运算符包括赋值运算符和递增、递减运算符，下面将讲解表达式的相关知识。

1．表达式的优先级

将运算符和操作数连接起来就形成了表达式，称为 Java 表达式，一个变量名和常量名都可以形成简单的表达式。Java 运算符有严格的优先级，它和数学中先乘除后加减的原则一样，按优先级从高到低进行运算，如表 2-9 所示。

表 2-9　运算符的优先级

优　先　级	运　算　符	解　　释
1	[ ]　()	分隔符
2	++　--　~ !	自增自减运算 按位取反 逻辑非
3	*　/　%	算术乘除运算
4	+　-	算术加减运算
5	>>　<<　>>>	移位运算
6	<=　<　>　>=	大小关系运算
7	==　! =	相等关系运算
8	&	按位与运算
9	^	按位异或运算
10	|	按位或运算
11	&&	布尔逻辑与运算
12	||	布尔逻辑或运算
13	? :	条件运算
14	=	赋值运算

2．表达式

下面将结合运算符讲解一些表达式，让读者加深对表达式的理解。看下面的一段代码：

```
public class Biaodashi1{
public static void main(String args[])
  {
        int a=11;
        int b=121;
        int c=25;
        double h=41.0;
             System.out.println(a-b+c+h*b);
  }
}
```

将上述代码编译并运行，得到如图 2-20 所示的结果。

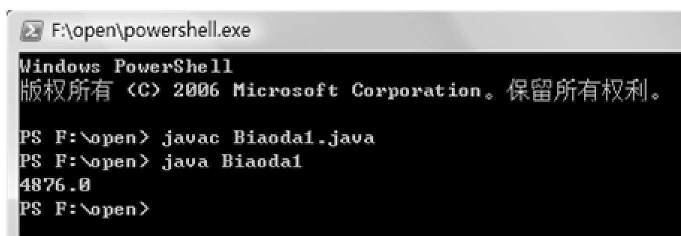


图 2-20　表达式

关系运算符是用来判断真假的，需要表达式来判断真假，结果为 true 或 false。下面将通

过一个例子，让读者认识关系运算符和表达式的综合运用知识，具体代码如下：

```
public class Biaoda2
{
   public static void main(String args[])
   {
   int a=231;
   int b=4;
    int h=56;
    int k=45;
    int x=a+h/b;
    int y=h+k;
  System.out.println(x);
  System.out.println(y);
  System.out.println(x==y);
}

}
```

将上述代码进行编译并运行，得到如图 2-21 所示的结果。

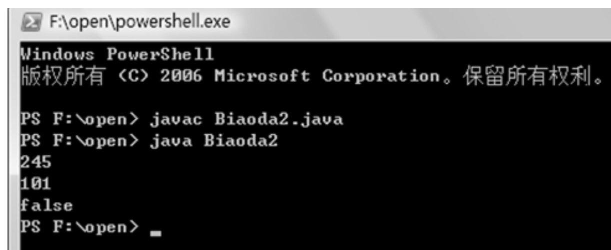


图 2-21　表达式

2.7　疑难问题解析

在本章的内容中，详细讲解了 Java 语法中变量、数据类型、运算符和表达式的基本知识。本节中，将对本章中比较难以理解的问题进行讲解。

读者疑问：在前面的一节中，讲解了注释，注释分为单行注释和多行注释，请问在什么情况下用单行注释，什么情况下用多行注释？

解答：这个没有严格的区别，单行的注释，一般都是简单的注释，多行注释一般都是需要注释许多内容。它没有严格的规定，用户只需要记住，注释是用来帮助程序理解的内容，它并不执行。

读者疑问：变量和常量是不是常常和数据类型有关系？听朋友讲过，只要变量和常量出现在程序中，都有一定的数据类型，这句话正确吗？

解答：变量和常量在程序中出现，程序员一定要定义它属于什么样的数据类型，如整

型、字符型等。如果不定义数据类型，这个常量和变量都会报错，导致程序不能运行，用户在使用变量和常量时，一定要定义数据类型。



职场点拨——扎实的基础很重要

如何才能成为一个优秀的程序员，这是很多初级读者比较关心的问题。其实这个问题很简单，无论是“开发人员”还是“测试人员”，一定要有扎实的基础作为后盾。究竟什么叫做扎实的基础呢?其实就是学习阶段所学的离散数学、算法与数据结构、操作系统、计算机体系结构、网络和编译原理，这些都是一个优秀程序员成长和前进的基础。

在面试时我经常听到这样的话：“我的理论和基础不行，但是让我写代码是没有问题的。”这样看来，基础真的没有用吗？笔者并不这样认为，读者可考虑以下 4 个问题。

1）如果不了解算法和数据结构，又如何从众多数据结构中做出最适合目前需求的选择呢？

2）如果您不了解操作系统，又如何能深入 Windows 系统，写好高级应用程序，或者在事后分析环境中做调试呢？

3）如果您不了解计算机体系结构，又如何能在多 CPU(多核)时代写出真正高效的应用程序呢？

4）编程是一门精密学科，编写的代码不能有一点错误，即使一个标点符号的错误都不能成功运行。即使成功运行，代码会符合健壮性的要求吗？这都是在学习基础知识时所必须掌握的，只有根基打得牢固，后面的高级开发才会水到渠成。

作为一名优秀的程序员，不光要求能写代码，而且要写好的代码，所以必须强调基础的重要性。很多程序员能力的高低，很可能就取决于这些基础。

第 3 章　程序控制结构

在 Java Web 程序中，程序控制结构是必不可少的，通过 Java 程序控制结构来控制 Java Web 工程是怎么执行的。程序通常是从第一行执行到最后一行，这自然无法满足程序设计的要求。为了满足程序设计的要求，在 Java 中引入了条件控制语句、循环语句和跳转语句，它们三者将颠覆程序顺序语句的执行方法，满足程序的设计需求。通过本章能学到如下知识。

- 条件控制语句。
- 循环。
- 跳转。
- 职场点拨——看职业的发展前景。

背景介绍

2008 年 X 月 X 日，周六，天气晴。

今天好累啊，刚参加了户外俱乐部的登山运动！一大早就跟着浩浩荡荡的队伍上山了。沿途中有很多岔路，经过一路疯狂拔高后，忽然看见前面的队伍停止了，我心中一阵窃喜，心想终于到达目的地了。可是上来一看，原来是迷路了，面前有 3 道岔路，向导也不知该走哪条路。



一问一答

小菜：“爬山好累啊，中途迷路了，当时感觉很害怕！”

Wisdom：“那你们怎么走出来的？”

小菜：“向导让我们在岔路口等候，他选了一条路走下去，走到一半发现错了，然后原路返回，然后继续走另外一条，发现又错了，原路返回，剩余那条就是正确的了！”

Wisdom：“排查法找路！你们选择路的过程还真像本章将要学的‘程序控制语句’！”

小菜：“程序控制语句？”

Wisdom：“对，爬山需要选择正确的道路，而 Java 程序执行时也需要选择将要执行哪条语句，这个选择过程就需要程序控制来实现！”

3.1　认识 Java 语句

在 Java Web 中，Java 程序并不是让程序单纯地顺序执行，还可以根据条件的不同执行

不同的程序代码。例如，既可以根据需要进行循环执行，也可以根据需要跳出执行。下面通过一段代码进行讲解，其代码（光盘：源代码/第 3 章/for1.java）如下：

```
public class for1
{
   public static void main(String args[])
   {
         for(int a=0;a<9;a++)
         {
               System.out.println(a);
         }
   }
}
```

将上面的代码编译并运行，得到如图 3-1 所示的结果。



图 3-1　执行结果

3.2　条件控制语句

在 Java Web 中，通常会使用 Java 程序里的条件语句来实现一些功能。条件语句通常分为两种，分别是 if 语句和 switch 语句。在执行条件控制语句过程中，程序会根据一个表达式的结果来判断执行哪一条语句，下面将详细讲解 if 语句和 switch 语句的基本用法。

3.2.1　if 语句

if 语句是多姿态的，它能以多种形式出现在程序中，实现各种功能。例如，有单个 if 语句、if-else 语句和 if-else-if 语句，下面进行详细讲解。

1. 单纯的 if 语句

一个简单的 if 语句结构十分简单，它是由保留字符 if、条件语句和后面的语句组成的。条件语句通常是一个布尔表达式，结果为 true 或 false。若条件为 true，则执行语句并继续处理其后的下一条语句；若条件为 false，则跳过该语句并继续处理紧跟着的整个 if 语句的下一条语句，其形式如下：

```
if（condition）statement1;
```

```
    statement2;
```

上述格式的执行流程如图 3-2 所示，条件为 true 时，则执行 statement1、statement2 语句；条件为 false 时，则只执行 statement2。

在上面的讲解中，读者认识了 if 语句的基本结构，也认识了 if 语句的执行方式。下面将通过一段实例代码认识 if 结构，其代码（光盘：源代码/第 3 章/ IfOne.java）如下：

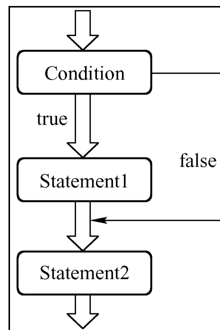


图 3-2　单纯的 if 语句

```
public class IfOne
{

   public static void main(String args[])
   {
         int a = 60;
         if(a>60){System.out.println("你及格了");
         }
         System.out.println("考查完毕，你需要补考");
   }

}
```

将上述代码编译并运行，得到如图 3-3 所示的结果。


F:\open\powershell.exe
Windows PowerShell
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

PS F:\open> javac IfOne.java
PS F:\open> java IfOne
考查完毕，你需要补考
PS F:\open>

图 3-3　简单的 if 语句

2．在 if 语句中加 else

if-else 语句是考虑到不同的情况，能够根据判定条件的真假来执行两种操作中的一种。使用 if-else 语句的基本格式如下：

```
if（condition）statement1;
else statement2;
```

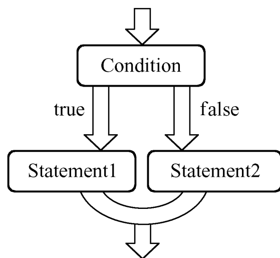


图 3-4　执行 if-else 语句

当条件 condition 为 true 时，执行 statement1；当 condition 为 false 时，则执行 statement2 语句，其流程图为 3-4 所示。

通过图 3-4 可以知道，在执行 if-else 语句时，不管用户输入的是什么值，经过 if-else 语句执行后都将会给出一个结果。下面通过一个具体实例来讲解 if-else 语句的基本用法。

实例 8：使用 if-else 语句进行判断

if-else 语句是优秀的条件语句，不管条件是不是满足，它都将给出一个答案，本实例的实现代码（光盘：源代码/第 3 章/IfElse1.java）如下：

```
public class IfElse1
 {

   public static void main(String args[])
   {
         int year = 2934;
         if(year>2940)
               {
               System.out.println("我的值大于 2940");
}
         else {
               System.out.println("我的值小于或等于 2940");
}
                     }

}
```

将上述代码进行编译并运行，得到如图 3-5 所示的结果。



图 3-5　if-else 语句



多学一招

通过上面的实例代码，使读者理解了 if-else 语句的基本用法。请读者再看下面的一段代码，看完后将程序进行编译并运行，然后查看运行结果如何。最后希望读者找出这段程序中的一个不足之处，其代码（光盘：源代码/第 3 章/IfElse2.java）如下：

```
public class IfElse2
{
   public static void main(String args[])
   {
         int hyear = 2048;
         if(hyear<2048)
               {
         System.out.println("我绝对相信我的值小于 2048");}
         else
{
```

```
                System.out.println("我的值大于或等于 2048");}
            }
    }
```

提示：不管是上面的实例，还是“多学一招”，这个代码都只能输出其中的一种情况，也就是说，程序对可能出现的两种情况进行判断，输出满足条件的一种情况。

3．最王道的 if 语句——if-else-if

if-else-if 语句实际上就是对 if 语句的嵌套，它可以对多种条件进行判断，形式如下：

```
if（condition1）
statement1;
else if（condition2）
statement2;
else statement3
```

首先它会判断第一个条件 condition1，当为 true 时执行 statement1，当为 false 时则继续执行下面的代码；当第二个条件 condition2 为 true 时执行 statement2；当 condition2 为 false 时则执行 statement3。具体流程如图 3-6 所示。

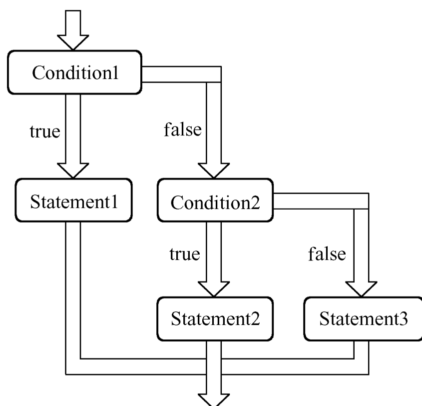


图 3-6　if-else-if 语句流程图

if-else-if 可以嵌套无限次，可以说只要遇到正确的 condition，就执行相关的语句，然后结束程序。

实例 9：使用 if-else-if 的嵌套

本实例的目的是让读者认识 if-else-if 控制语句的用法，具体代码如下：

```
public class IfIf1 {
   public static void main(String args[])
   {
         int yonth = 3;
         if(yonth<20)
            System.out.println("十分年轻，正是学习的年龄");
         else if(yonth<30)
```

```
                System.out.println("正当壮年，好好干一番");
            else if(yonth<45)
                System.out.println("而立之年");
            else if(yonth<60)
                    System.out.println("已经老了");
            else
                System.out.println("垂垂老去");
            }
    }
```

将上述代码编译并运行，得到如图 3-7 所示的结果。

```
F:\open\powershell.exe
Windows PowerShell
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

PS F:\open> javac IfIf1.java
PS F:\open> java IfIf1
十分年轻，正是学习的年龄
PS F:\open> _
```

图 3-7　if 语句的嵌套


多学一招

if 语句嵌套实际上是对多种条件的判断，不管条件有几种情况，都可以通过 if 语句进行判断，例如，下面的代码（光盘：源代码/第 3 章/IfIf2.java）有多种情况:

```
    public class IfIf2
    {
        public static void main(String args[])
        {
            int chengji = 95;
            if(chengji>90)
                System.out.println("十分优秀");
            else if(chengji>70)
                System.out.println("成绩一般");
            else if(chengji>=60)
                System.out.println("及格了，很幸运");
            else
                System.out.println("不及格，惨了，需要重修");
            }
    }
```

4．轻松书写漂亮的 if 语句

if 语句是条件控制语句，它的语法格式很容易理解，但是在具体使用时需要注意一些问题，下面通过一段代码来讲解它的用法，其代码（光盘：源代码/第 3 章/ifyou1.java）如下：

```
    public class ifyou1
```

```
{
   public static void main(String args[])
   {
         int chengji = 55;
         if(chengji>=60)
            System.out.println("及格");
            System.out.println("考核完毕");

   }
}
```

它会输出什么样的结果？很多初学者会觉得不会有啥结果，实际上呢？编译并运行后会得到如图 3-8 所示的结果

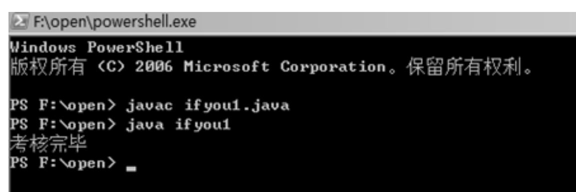


图 3-8　不正规编写 if 语句

在上述代码中没有正规地编写 if 语句，if 语句只对下面一句生效，对后面的都不生效，再看笔者修改后的代码，其代码（光盘：源代码/第 3 章/ifyou2.java）如下：

```
public class ifyou2
{
   public static void main(String args[])
   {
         int chengji = 45;
         if(chengji>=60)
         {
               System.out.println("及格");
               System.out.println("考核完毕");
         }
   }
}
```

在上述代码中使用了标准的 if 语句块，只有满足条件后，它才能执行括号里的内容。在上面的程序中成绩变量等于 45，并不满足成绩大于 60 的情况，所以它并不执行，执行结果应该为空。下面来编译这段程序并执行，会得到如图 3-9 所示的效果。

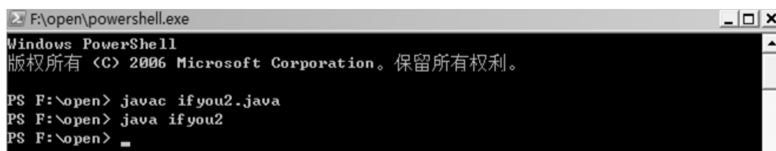


图 3-9　没有任何执行效果

提示：在书写 if 语句时，一定要编写规范，多使用大括号，就算只有一个语句，也要使用大括号，只有这样，程序才会容易懂，不容易出现逻辑错误。

3.2.2 switch 语句

虽然 switch 语句和 if 嵌套语句十分类似，但是在处理一些问题时，if 嵌套语句没有 switch 语句优越。使用 switch 的格式如下：

```
switch（expression）
{
Case value1：statement1；
break；

case value2：statement2；
break；
case value3：statement3；
break；
default：statement4；
}
```

在上面的格式中，关键字 expression 必须是 byte、short、int 和 char 类型，每个值必须是与 expression 类型兼容的一个常量，而且不能重复。switch 控制语句的运作流程如图 3-10 所示。

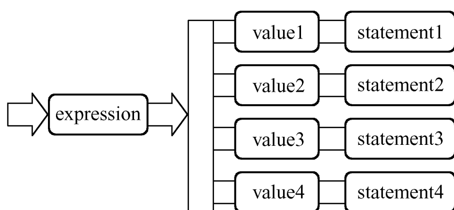


图 3-10　switch 语句

1．switch 语句的一般情况

switch 语句的一般情况是最简单的语句，下面通过一段代码进行讲解，其代码（光盘：源代码/第 3 章/switch1.java）如下：

```
public class switch1
{
  public static void main(String args[])
  {
      int a=7;
      switch(a)
      {
          case 1:
            System.out.println("a=1");
            break;
```

```
                case 2:
                   System.out.println("a=2");
                   break;
                case 3:
                   System.out.println("a=3");
                case 7:
                         System.out.println("a=7");
                   break;
                default:
                   System.out.println("no");
          }
    }
}
```

上面的这段代码很简单，当执行 switch 语句时会一直寻找 a=7 这个条件，当找到后会输出 a=7。将上述代码进行编辑并运行，得到如图 3-11 所示的结果。

```
F:\open\powershell.exe
Windows PowerShell
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

PS F:\open> javac switch1.java
PS F:\open> java switch1
a=7
PS F:\open>
```

图 3-11　一般的 switch 语句

2．break 与 switch

在 switch 语句中可以没有 break，但是如果使用了 break 关键字，switch 就不再执行了。下面通过一段代码进行讲解，其代码（光盘：源代码/第 3 章/switch2.java）如下：

```
public class switch2
{
   public static void main(String args[])
   {
         int month = 12;
         switch(month)
         {
               case 12:
               case 1:
               case 2:
                  System.out.println("冬季");
                  break;
               case 3:
               case 4:
               case 5:
                  System.out.println("春季");
                  break;
```

```
                case 6:
                case 7:
                case 8:
                    System.out.println("夏季");
                    break;
                case 9:
                case 10:
                case 11:
                    System.out.println("秋季");
                    break;
                default:
                    System.out.println("输入错误");
            }
        }
    }
```

将上述代码编译并运行，得到如图 3-12 所示的结果。

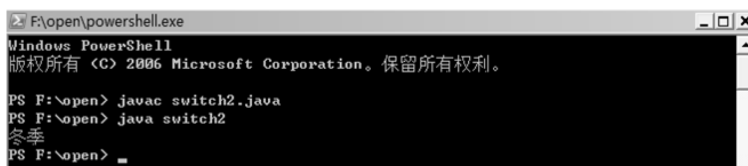


图 3-12　执行结果

3．default 不一定只在 switch 的末尾

default 语句是一个很重要的关键字，对于 switch 语句来说更加重要。default 语句一般位于 switch 的末尾，但是它并不是说只能在 switch 语句的末尾，它可以在 switch 的任意位置，例如，下面的代码（光盘：源代码/第 3 章/switch3.java）：

```
public class switch3
{
        public static void main(String args[])
        {
            int a=2997;
            switch(a)
            {
                case 2992:
                    System.out.println("a=2992");
                default:
                    System.out.println("no");
                case 1997:
                    System.out.println("a=2997");
                case 2008:
                    System.out.println("a=2008");
            }
```

```
                }
        }
```

将上述代码进行编译并运行，得到如图 3-13 所示的结果。

```
F:\open\powershell.exe
Windows PowerShell
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

PS F:\open> javac switch3.java
PS F:\open> java switch3
no
a=2997
a=2008
PS F:\open> _
```

图 3-13　default 关键字

3.3　循环语句

在 Java Web 中，可以使用 Java 语言中的循环语句来实现循环功能。在循环语句中最为常见的循环有 3 种，分别是 for 循环、while 循环和 do-while 循环。这 3 种循环的使用方法基本相同，只是细节稍有变化。下面将一一讲解上述循环的基本知识。

3.3.1　for 循环语句

for 循环是一个功能强大并且使用方便的循环语句，是 Java 程序中最为常见的循环语句，它的基本格式如下：

```
        for（initialization;condition;iteration）
        {
        }
```

for 循环语句由 3 个部分组成，分别是变量的声明和初始化、布尔表达式以及循环表达式，每一部分都用分号分隔。在执行 for 循环的过程中，循环启动后，最先开始执行的是初始化部分（求解表达式 1），然后执行布尔表达式（表达式 2）的值，如果符合条件则执行循环，不符合条件则跳出循环。

下面看上述三部分的具体介绍：

- 声明和初始化：for 语句中的第一部分是关键字 for 之后的括号内的声明和初始化变量，声明和初始化发生在 for 循环内任何操作前，声明和初始化只在循环开始的时候发生一次。
- 条件表达式：执行的下一部分是条件表达式，它的计算结果必须是布尔值，在 for 循环中，只能有一个表达式。
- 循环表达式：在 for 循环体每次执行后，都执行循环表达式，它设置该循环在每次循环之后要执行的操作，它永远在循环体运行后执行，也就是最后执行。

for 循环的执行流程也十分简单，通过图 3-14 所示的过程，读者就可以很清楚地理解 for 循环语句是如何执行的。

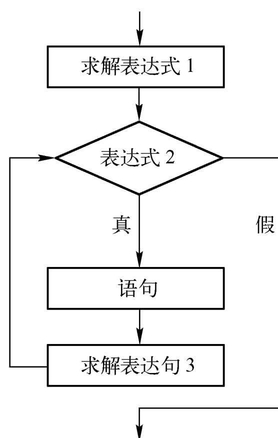


图 3-14　for 语句的执行流程

实例 10：使用 for 循环语句

本实例的目的是让读者更加深入理解 for 循环语句的用法，其代码（光盘：源代码/第 3 章/forone1.java）如下：

```
public class forone1
 {
   public static void main(String args[])
   {
         for(int j=0;j<100;j++)
         {
               System.out.print(j);
         }
         }
}
```

将上述代码编译并运行，得到如图 3-15 所示的结果。



图 3-15　for 循环语句



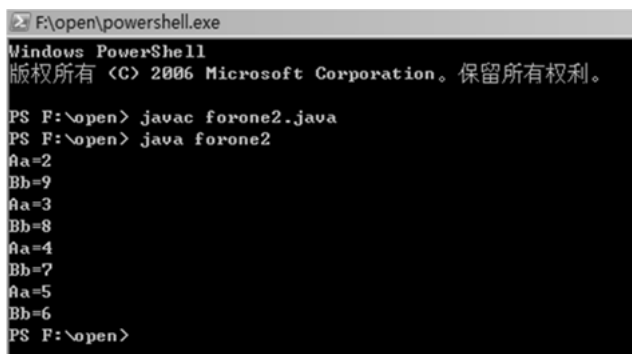
多学一招

除了上面的循环语句，在 for 循环语句中还可以允许有多个控制循环。Java 允许在 for

循环的初始化部分和循环表达式部分声明多个变量，每个变量用逗号隔开，例如，下面的代码（光盘：源代码/第 3 章/ forone2.java）：

```
public class forone2
{
   public static void main(String args[])
   {
         for(int Aa=2,Bb=9;Aa<Bb;Aa++,Bb--)
         {
              System.out.println("Aa="+Aa);
              System.out.println("Bb="+Bb);
         }
         }
}
```

将上述代码编译并运行，得到如图 3-16 所示的结果。


F:\open\powershell.exe
Windows PowerShell
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

PS F:\open> javac forone2.java
PS F:\open> java forone2
Aa=2
Bb=9
Aa=3
Bb=8
Aa=4
Bb=7
Aa=5
Bb=6
PS F:\open>

图 3-16　for 循环语句

3.3.2　while 循环语句

while 循环语句是 Java 控制语句中的另外一种循环语句，当不知道循环多少次，就可以使用 while 循环语句。当 while 循环语句块结果为真时，while 语句会重复执行一条语句和语句块。while 循环语句的格式如下：

```
While (condition)
{
}
```

while 循环语句的执行流程如图 3-17 所示。

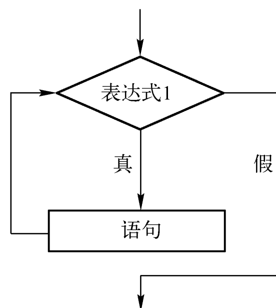


图 3-17　while 循环控制语句的流程图

在图 3-17 所示的流程图中，值得注意的是表达式 1（条件表达式）是布尔表达式，只要表达式 1 为真，语句块将被执行，当表达式为假时，它将跳出循环。

实例 11：使用 while 循环语句实现累加

本实例的目的是让读者加深理解 while 循环语句的用法，其代码（光盘：源代码/第 3 章

/whileone1.java）如下：

```
public class whileone1
 {

   public static void main(String args[])
   {
         int a=0;
         while(a<10)
         {
              System.out.println(a);
              a++;
         }
   }
}
```

将上述代码编译并运行，得到如图 3-18 所示的结果。

```
F:\open\powershell.exe
Windows PowerShell
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

PS F:\open> javac whileone1.java
PS F:\open> java whileone1
0
1
2
3
4
5
6
7
8
9
PS F:\open>
```

图 3-18　while 语句



多学一招

在实例 11 的 while 循环语句中，只是展示了 while 语句一个简单的循环过程。除此之外，while 语句还常常用来累加一些数据，如网页访问的数量等，请看下面的一段代码（光盘：源代码/第 3 章 whileone2.java）：

```
public class whileone2
{
   public static void main(String[] args)
{
         int sum=0;
         int i=1;
//由于是计算自然数，所以 1 的初始值设置为 1
```

```
            System.out.println("累加和不大于 10 的所有自然数如下：");
            while(sum<10)
    {
                sum=sum+i;
                System.out.println(i);
                i++;      //该语句一定不能少
            }
        }
    }
```

将上述代码编译并执行，得到如图 3-19 所示的结果。



图 3-19　while 语句

3.3.3　do…while 循环语句

初学者可以这么理解 do…while 循环语句：在某种情况下，即使条件为假时，也需要执行语句一次，在执行一次循环后再测试表达。由此可见，Java 中 do-while 语句的最大特点是，至少会执行一次循环体，因为它的条件表达式在循环的最后。do-while 循环语句的格式如下：

```
do
{
}
While（condition）;
```

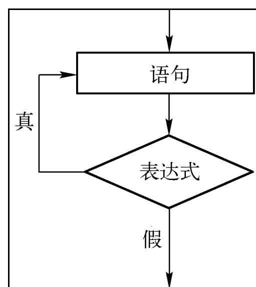


图 3-20　do-while 循环语句

do-while 的运行流程如图 3-20 所示。



在书写 do-while 程序时，千万不要忘记 while()语句后面有一个“;”,初学者容易漏掉这个分号，这样会造成编译和运行时报错。

实例 12：使用 do-while 循环语句

本实例的目的是让读者深刻理解循环语句 do-while 的运作流程：先执行一次，然后判断表达式，如果表达式为真，则循环继续，如果表达式为假，循环到此结束。实例代码（光盘：源代码/第 3 章/doone1.java）如下：

```
public class doone1
```

```
    {
       public static void main(String[] args)
       {
       int tao=2;
       do
       {
       tao=tao-5;
       System.out.println(tao);
       }

       while (tao>=5);
       }
    }
```

将上述代码进行编译并运行，得到如图 3-21 所示的结果。

```
F:\open\powershell.exe
Windows PowerShell
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

PS F:\open> javac doone1.java
PS F:\open> java doone1
-3
PS F:\open>
```

图 3-21　do-while 语句



多学一招

do-while 语句也同样具有 for 循环语句和 while 循环进行累加的功能，其代码（光盘:源代码/第 3 章/doone2.java）如下:

```
    public class doone2
     {
       public static void main(String args[])
       {
             int i = 1;
             int sum = 0;
             do
             {
             sum += i++;
             }
             while(i<=100);
             System.out.println(sum);
       }
    }
```

将代码编译并运行，得到如图 3-22 所示的结果。

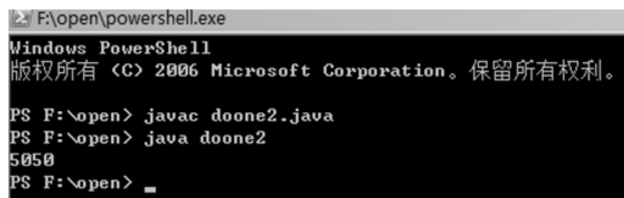


图 3-22　do-while 循环语句的累加

3.3.4　循环语句的嵌套

在 Java 程序中，因为在使用嵌套循环语句时一般都是使用 for 循环语句，所以在此也只讲解 for 循环语句的嵌套用法。所谓 for 嵌套语句，实际上就是指两个或者两个以上的 for 语句组合的语句。

实例 13：使用 for 嵌套语句

本实例演示了使用 for 嵌套语句的用法，本实例能够通过嵌套语句打印菱形。具体代码（光盘：源代码/第 3 章/forqian1.java）如下：

```
public class forqian1
{
   public static void main(String[] args)
   {
   int n = 7;
   int o = (n/2);
   int t= 1;
   int step = 2;
   for (int i=0; i<n; i++)
   {
   for (int j=0; j<Math.abs(o); j++)
   {
   System.out.print((char)32);
   }
   o--;
   for (int k=1; k<=t; k++)
   {
   System.out.print("*");
   }
   t = t + step;
   if (t == n)
   {
   step = -step;
   }
   System.out.println();
   }
   }
}
```

将上述代码进行编译并运行，得到如图 3-23 所示的结果。

```
F:\open\powershell.exe
Windows PowerShell
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

PS F:\open> javac forqian1.java
PS F:\open> java forqian1
   *
  ***
 *****
*******
 *****
  ***
   *
PS F:\open>
```

图 3-23　for 语句的嵌套


多学一招

通过 for 语句的嵌套可以实现很多与众不同的功能，上面实例打印了一个菱形，在下面的代码中，通过 for 语句的嵌套组成了一个“倒置三角形”，其代码（光盘：源代码/第 3 章/forqian2.java）如下：

```
public class forqian2
{
   public static void main(String[] args)
    {
     for(int i=9; i>=1; i--)
     {
      for(int j=i; j>=1; j--)
      {
      System.out.print(j + " ");
       }
       System.out.println();
     }
    }

}
```

将代码编译并运行，得到如图 3-24 所示的结果。

```
F:\open\powershell.exe
Windows PowerShell
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

PS F:\open> javac forqian2.java
PS F:\open> java forqian2
9 8 7 6 5 4 3 2 1
8 7 6 5 4 3 2 1
7 6 5 4 3 2 1
6 5 4 3 2 1
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
PS F:\open> _
```

图 3-24　嵌套语句

3.4　跳转语句

在 Java Web 程序设计中，通过使用 Java 跳转语句来实现跳转功能，目的是控制循环语句或者条件语句转移到其他循环语句或程序的其他部分。在 Java 中主要有 3 种跳转语句，分别是 break 语句、continue 语句和 return 语句，本节将对上述跳转语句进行详细讲解。

3.4.1　break 跳转语句

在前面讲解 switch 语句时曾经提到过 break 语句，用它可以终止一个语句。除此之外，break 语句还可以退出循环。另一个重要作用是设置程序“到哪里去”，相当于英文中的 goto。break 语句可以分为无标号退出循环和有标号退出循环两种，下面将详细讲解这两种用法。

1．无标号退出循环

无标号退出循环就是直接退出循环，当在循环语句中遇到 break 语句后，循环就会立即终止，程序在循环体外面的语句将会重新开始。

实例 14：使用 break 实现无标号退出循环

本实例的具体实现代码（光盘：源代码/第 3 章/breakone1.java）如下：

```
public class breakone1
{
   public static void main(String args[])
   {
         /**
          * for 语句中的 break
          */
         for(int d=0;d<9;d++)
         {
               if(d==2)
               {
                     break;
               }
               System.out.println(d);
         }
         /**
          * while 语句中的 break
          */
         int i=0;
         while(i<9)
         {
               if(i==2)
               {
                     break;
               }
```

```
                System.out.println(i);
                i++;
            }
            /**
             * do-while 语句中的 break
             */
            int j=0;
            do
            {
                if(j==2)
                {
                    break;
                }
                System.out.println(j);
                j++;
            }
            while(j<9);
    }
    }
```

将上述代码编译并运行，得到如图 3-25 所示的结果。

```
powershell
Windows PowerShell
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

PS F:\> cd open
PS F:\open> javac breakone1.java
PS F:\open> java breakone1
0
1
0
1
0
1
PS F:\open> _
```

图 3-25　break 语句的用法


多学一招

break 语句除了上述实例中演示的用法外，还可以用在嵌套语句中来终止它所在的循环，例如，下面的代码（光盘：源代码/第 3 章/breakone2.java）：

```
    public class breakone2 {
       public static void main(String args[])
       {
            /**
             * break 出现在内循环
             */
            for(int i=1;i<10;i++)
            {
```

```
            for(int j=i;j<10;j++)
            {
                if(j==5)
                {
                    break;
                }
                System.out.print("☆");
            }
            System.out.println();
        }
        /**
         * break 出现在外循环
         */
        for(int a=1;a<10;a++)
        {
            if(a==5)
            {
                break;
            }
            for(int b=a;b<10;b++)
            {
                System.out.print("☆");
            }
            System.out.println();
        }
    }
}
```

将上述代码编译并运行，得到如图 3-26 所示的结果。

```
powershell
PS F:\open> javac breakone2.java
PS F:\open> java breakone2
☆☆☆☆
☆☆☆
☆☆
☆

☆☆☆☆
☆☆☆
☆☆
☆
☆☆☆☆☆☆☆☆☆
☆☆☆☆☆☆☆☆
☆☆☆☆☆☆☆
☆☆☆☆☆☆
PS F:\open>
```

图 3-26　break 用在循环嵌套

2．有标号退出循环

有标号退出循环只能使用在嵌套的循环语句中。在嵌套的循环语句中，可以在循环语句前面加一个标号，在使用 break 语句时，就可以使用 break 后面紧接着一个循环语句前面的标号来退出该标号所在的循环了。例如，下面的代码（光盘：源代码/第 3 章/youbiao.java）就演示了有标号退出循环这种用法。

```
public class youbiao {
  public static void main(String args[])
  {
        out:for(int a=0;a<10;a++)
            {
              System.out.println("a="+a);
              for(int b=0;b<10;b++)
              {
                    if(b==5)
                    {
                          break out;
                    }
                    System.out.println("b="+b);
              }
            }
  }}
```

将上述代码编译并运行，得到如图 3-27 所示的结果。

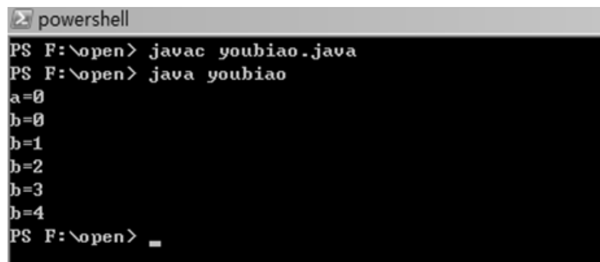


图 3-27　有标号退出循环



带标号的 break 语句只能放在这个标号所指的循环里面，如果放到别的循环体里面就会出现编译错误。

3.4.2　continue 语句

continue 语句是一个很“霸道”的跳转语句，它的作用就是强制一个循环提前返回，换句话说，让循环继续执行，但不执行本次循环剩余的循环体中的语句。

实例 15：使用 continue 语句

本实例演示了使用 continue 语句的方法，其代码（光盘：源代码/第 3 章/continueone1.java）如下：

```
public class continueone1
{
   public static void main(String args[])
   {
         for(int a=0;a<19;a++)
         {
               System.out.print(a);
               if(a%3==0)
               {
                     continue;
               }
               System.out.println("￥");
         }
   }
}
```

将上述代码编译并运行，得到如图 3-28 所示的结果。

```
powershell
PS F:\open> javac continueone1.java
PS F:\open> java continueone1
01￥
2￥
34￥
5￥
67￥
8￥
910￥
11￥
1213￥
14￥
1516￥
17￥
18PS F:\open>
```

图 3-28　continue 语句



多学一招

除了实例 15 中的用法外，continue 语句和 break 语句一样，也存在有标号的 continue 语句，在下面九九乘法表的实现代码中就使用了有标号的 continue 语句，其代码（光盘：源代码/第 3 章/continueone2.java）如下:

```
public class continueone2
{
   public static void main(String args[])
   {
         out:for(int a=1;a<=9;a++)
             {
               for(int b=1;b<=9;b++)
               {
                     if(b>a)
```

```
                    {
                          System.out.println();
                          continue out;
                    }
                    System.out.print(" "+a*b);
              }
              System.out.println();
           }
    }
 }
```

将上述代码编译并运行，得到如图 3-29 所示的结果。

```
powershell
PS F:\open> javac continueone2.java
PS F:\open> java continueone2
 1
 2 4
 3 6 9
 4 8 12 16
 5 10 15 20 25
 6 12 18 24 30 36
 7 14 21 28 35 42 49
 8 16 24 32 40 48 56 64
 9 18 27 36 45 54 63 72 81
PS F:\open> _
```

图 3-29　有标记的 continue 语句

3.4.3　return 语句

return 语句的功能是返回一个方法，并把控制权交给调用它的语句，其形式如下：

```
return[表达式];
```

表达式是可选参数，表示要返回的值，它的数据类型必须同方法声明中的返回值类型一致，这可以通过强制类型转换实现。

下面通过一段代码讲解 return 语句的用法，其代码（光盘：源代码/第 3 章/returnone1.java）如下：

```
public class returnone1
 {
   public static void main(String[] args)
   {
              Cat c1 = new Cat(1,2,3); //实例化
              Cat c2 = new Cat(1,2,3);
              System.out.println(c1.equals(c2));
              }
              }
              class Cat {
              int colour;
```

```
                    int height,weight;
                    public Cat(int colour,int height,int weight) {
                    this.colour = colour;  //使用全局变量
                    this.height = height; //使用全局变量
                    this.weight = weight; //使用全局变量
                    }
                    public boolean equals(Object obj) {
                    if (obj==null) return false;
                    else
                    if(obj instanceof Cat) {
                    Cat c = (Cat)obj;
                    if((c.colour==colour)&&(c.height==height)&&(c.weight==weight)) {
                    return true;
                    }
                    }
                    return false;
                    }
        }
```

将上述代码编译并运行，得到如图 3-30 所示的结果。

```
powershell
PS F:\open> javac returnone1.java
PS F:\open> java returnone1
true
PS F:\open> _
```

图 3-30　return 跳转语句

提示：return 语句一般放在方法的最后，用于退出当前语句并返回一个值。当把单独的 return 语句放在一个方法的中间时，会产生编译错误，其错误为“Unreachable code”。读者可以通过将 return 语句用 if 语句括起来的方法，将 return 语句放在一个方法中间，用来实现退出在程序中未执行完方法中的全部语句。


多学一招

除了上述实例中演示的返回一个布尔值的用法外，return 还可以返回一个数值，代码（光盘：源代码/第 3 章/returnone2.java）如下:

```
        public class returnone2
        {
          public static int gcd(int a, int b) {
                int min = a;
                int max = b;
                if (a > b)
        {
                min = b;
                max = a;
```

```
            }
            if (min == 0)
            return max;
            else
            return gcd(min, max - min);
            }
        public static void main(String[] args) {
        System.out.println(returnone2.gcd(26, 48));
        }
    }
```

将上述代码进行编译并运行，得到如图 3-31 所示的结果。

```
powershell
PS F:\open> javac returnone2.java
PS F:\open> java returnone2
2
PS F:\open>
```

图 3-31　返回数值的 return 语句

3.5　疑难问题解析

在本章的内容中，详细讲解了 Java 语言中流程控制语句的基本知识。本节中，将对本章中比较难理解的问题进行讲解。

读者疑问：在本章中，已经不再是前面讲解的顺序语句了，引进了条件控制语句、循环语句、跳转语句等一系列语句，它们三者之间有联系吗？

解答：对，程序的顺序是默认的，在很多情况不适用，所以就有了本章所讲的控制结构语句，这三者之间是互相依靠的，尤其是跳转语句和循环语句、跳转和条件语句，它们的搭配使用在程序中最为常见，单独的一种控制结构语句，起的作用很小，功能很弱，只有通过它们之间的配合，才能使功能强大。

读者疑问：在前面的循环语句中，讲解了循环嵌套，在本章中，只见到了 for 循环嵌套，其他两种循环不可以嵌套吗？

解答：while 语句和 do-while 语句也可以嵌套，只不过使用最多的是 for 语句嵌套，而且 while 语句和 do-while 语句嵌套，相对来说较为复杂，如果读者熟练掌握了 for 语句的嵌套后，不妨去看看资料，查看一下 while 语句和 do-while 语句嵌套的问题。



职场点拨——看职业的发展前景

人的一生会面临很多选择，并且不同的时期所面临的选择也不会相同。对于广大读者来

说，当学业有成后，如果选择了一个未来发展前景好的职业，以后的职业发展会十分顺畅；反之则可能在今后的发展中需要职业转型，需要付出更多努力。

（1）选择的重要性

职业选择，是职业人生的头等大事，也是人一生之中的几件大事之一。一般来说，面临职业选择的人群有如下两类。

- 第一类是刚刚毕业走上社会的大学生及研究生。
- 第二类就是在社会上打拼过几年，第一次选择的职业不适合自己或者未来发展前景不好的人群。

不管是刚刚毕业的学生，还是面临职业转型的人群，他们需要共同面临的问题就是该选择什么样的职业，怎样规划自己的职业人生。

（2）前景很重要

人们在面临重要选择的时候，往往只看到眼前利益。职业选择作为人生的重大选择之一，一定要看好发展前景。很多人在求职时，并没有对自己进行职业定位，考虑的多是单位入职后的基本待遇。从职业规划的角度考虑，应该特别慎重选择第一份工作，因为第一份工作将对个人的职业态度、职业习惯产生深远的影响，并且也决定了未来的发展高度。当工作几年之后，如果发现兴趣爱好、个人专长都与目前职业不匹配，想跳槽谋高薪高职也只能在目前行业中选择，这就出现了职业发展瓶颈，需要寻求职业转型。

（3）一个例子

在当前的开发公司中，管理软件应用顾问这一职业前景被业界看好，成为 IT 从业人群和企业管理人员转型的首选职业。随着信息化行业的快速发展，管理软件应用顾问的人才需求还会不断增加，并且这个增加是双向的，一方面是管理软件厂商需要实施顾问和售前顾问等去帮助企业应用信息化，一方面是企业信息化需要更多的信息化人才，这也是项目成功与否和管理持续优化的关键因素。有相关行业经验的顾问可直接在软件公司做 CIO 或信息化项目负责人。

在企业中做顾问，通过在实践项目中历练，了解熟悉企业业务，会很快成为企业业务部门的骨干。所以，从某种程度来说，实施项目组是为企业各个业务部门培养人才的蓄水池。

（4）结论

由此可见，在职业发展的过程中，职业选择是第一位的。有了正确的、合适的选择，我们的努力才有价值，职业发展才会顺畅。在选择职业时切勿急功近利，避免“捡了芝麻，丢了西瓜”。

第 4 章　数　　组

在 Java Web 开发应用中，经常会使用 Java 语言中的数组。有了数组，就可以用相同名字引用一系列变量，并用数字（索引）来识别它们。在编程中使用数组后，可以缩短和简化程序。并且可以利用索引值设计一个循环，实现高效处理。Java 数组有上界和下界，数组的元素在上下界内是连续的。在本章的内容中，将详细讲解 Java 数组和对数组进行基本操作的知识。通过本章能学到如下知识。

- 一维数组。
- 二维数组。
- 多维数组。
- 操作数组。
- 职场点拨——学编程的误区。

背景介绍

2008 年 X 月 X 日，天气阴转晴

今天，我准备开始学习数组。我很不明白，已经有了变量和常量了，还推出数组有什么用？我决定明天向表哥请教。



一问一答

小菜："Java 中的数组究竟有什么作用，不是已经有变量和常量了吗？"

Wisdom："呵呵，当时我在学习这部分内容时也有这个疑问。现在我明白了，数组是为了实现某种用途，必须将变量和常量等数据包装一下，以一种新的面貌展现在我们面前。"

小菜："数组很重要吗？"

Wisdom："当然重要了！数组是 Java 的最重要组成部分，数组属于构造数据类型。一个数组可以分解为多个数组元素，这些数组元素可以是基本数据类型或是构造类型。因此按数组元素的类型不同，数组又可分为数值数组、字符数组、指针数组、结构数组等各种类别。"

4.1　简单的一维数组

在 Java 数组中，一维数组是最简单的数组。所谓一维数组，实质上就是一组相同类型

的数据的线性集合，当在程序中需要处理一组数据时，或者传递一组数据时，可以应用这种类型的数组。在本节的内容中，将详细讲解一维数组的基本知识。

4.1.1　声明一维数组

在讲解之前，先举一个例子。假设在一个公司，财务部门需要对公司员工工资进行处理，如计算平均工资、最高工资等。假设公司有 650 名员工，根据前面所学的知识，需要声明 650 个变量。在编写的过程中，声明这么多变量的工作量是巨大的。其实不必这么麻烦，只需声明一个数组就可以轻松解决这样的问题。

数组就是元素的集合体，每一个元素都拥有一个索引值，只需要指定索引值就可以取出对应的数据，声明一维数组的基本格式如下。

```
int[] array;
```

或者：

```
int array[];
```

虽然上述两种声明形式不同，但是含义是一样的，各个参数的具体说明如下。

- int：数组元素类型。
- array：数组名称。
- []：一维数组的内容都是通过这个符号括起来。

除了上面的声明整型数组外，还可以声明多种数据类型的数据，例如：

```
boolean[] array;//声明布尔数组
float[] array;//声明浮点数组
double[] array;//声明双精度数组
```

4.1.2　数组的创建和赋值

创建数组实质上就是为数组申请存储空间，数组的创建需用大括号（{}）括起来，然后将一组相同类型的数据放在存储空间里，Java 编译器负责管理和分配存储空间。创建一维数组格式如下。

```
int[] A={2,4,6,7,8,9,10};
```

要想完全理解上述数组，请看图 4-1 所示的内部结构图。

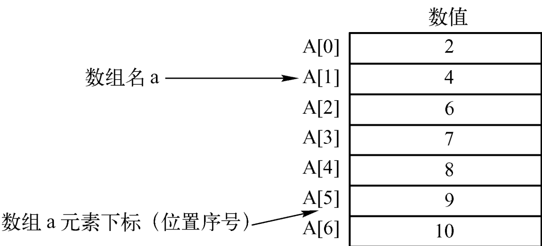


图 4-1　一维数组内部结构

在图 4-1 中，A 为数组名称，方括号的值为序号即下标。A 数组的第一个值就用 A[0]表示，第 2 个值就用 A[1]表示，依此类推。上面一节中讲解的公司员工工资的问题，可以使用这种数组方法来解决，即可以用 A[0]、A[1]、A[2]、A[3]…A[649]来表示每个员工。

实例 16：实现一维数组的打印

本实例的功能是创建一个一维数组，然后打印输出一维数组的数据，具体实现代码（光盘：源代码/第 4 章/shuzu1.java）如下。

```
public class shuzu1
{
   public static void main(String[] args)
   {
         int[] a1={12,13,14,17,18,19};
         int[] a2;
         a2=a1;
         for(int i=0;i<a1.length;i++)
         {
               a2[i]++;
               System.out.println("a1["+i+"]="+a1[i]);
               System.out.println("a2["+i+"]="+a2[i]);
         }
   }
}
```

将上述代码进行编译并运行，得到如图 4-2 所示的效果。

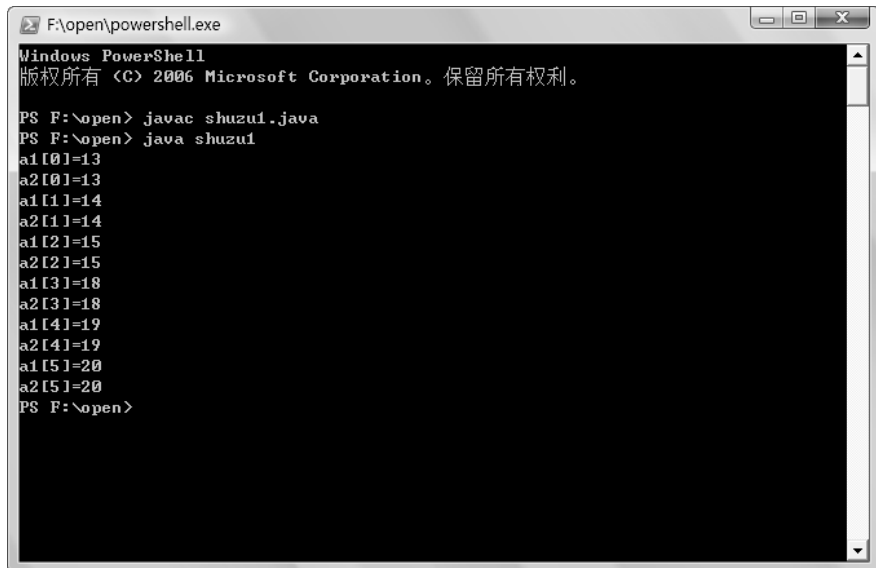


图 4-2　数组

提示：在上面的数组中，读者需要注意数组计数都是从零开始的，最大数组下标为“length-1”，在程序中，数组 a2 没有任何元素，它此时只是被实例化了一个对象，告诉编译

器为它分配一定的存储空间，然后数组 a1 赋值给 a2，这个编译操作实际上就是将 a1 数组的内存地址赋给数组 a2。


多学一招

在上面的实例中，没有给数组 a2 赋值。再看在下面的代码中，将两个数组都赋予了值，读者请仔细阅读程序，给出一个运行结果。然后将代码复制到自己的电脑中，进行编译并运行，此时对比自己的答案与编译结果看是否一致。其代码（光盘：源代码/第 4 章/shuzu2.java，如下。

```
public class shuzu2
{
public static void main(String[] args)
   {
                  int[] a1={112,113,114,117,118,119};
        int[] a2={1,5,11,12,14,15};
        a2=a1;
        for(int i=0;i<a1.length;i++)
        {
             a2[i]++;
             System.out.println("a1["+i+"]="+a1[i]);
             System.out.println("a2["+i+"]="+a2[i]);
        }
   }
}
```

将上述代码编译并运行后，会发现 a1 与 a2 数组的值完全相同，效果如图 4-3 所示。

```
F:\open\powershell.exe
Windows PowerShell
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

PS F:\open> javac shuzu2.java
PS F:\open> java shuzu2
a1[0]=113
a2[0]=113
a1[1]=114
a2[1]=114
a1[2]=115
a2[2]=115
a1[3]=118
a2[3]=118
a1[4]=119
a2[4]=119
a1[5]=120
a2[5]=120
PS F:\open> _
```

图 4-3　数组的赋值

4.1.3　对数组进行初始化

数组是一个对象，它和前面讲解的基本数据类型相同，用户是可以对其进行初始化的，在初始化的时候，用户可以规定数组的大小，也可以初始化数组中每一个元素，下面讲解三种初始化数组的格式。

```
int[] a=new int[17];
int[] a=new int{1,2,3,4,5,6,7,8};
int[] a={1,2,3,4};
```

参数介绍如下。

- int：数组类型。
- a：数组名称。
- new：对象初始化语句。

学过 C++的人知道，第 3 种初始化格式是 C++初始化数组的方法，但是 Java 也支持。读者在编写程序的时候，可根据习惯选择这三种初始化的一种。

实例 17：初始化一维数组实现打印数据

在本实例中，首先初始化了一个一维数组，然后打印输出数组内的数据，具体实现代码（光盘：源代码/第 4 章/ shunew.java）如下。

```
import java.util.Random;//插入 Random 包
public class ShuNew
{
  public static void main(String[] args)
{
                  Random rand=new Random(); //实例化 Random 类对象
int[]a=new int[rand.nextInt(20)];//随机产生 0～20 之间的数作为 int 数组的长度
double[]b=new double[rand.nextInt(20)];
//随机产生 0～20 之间的数作为 int 数组的长度
        System.out.println("a 的长度为"+a.length);
        System.out.println("b 的长度为"+b.length);
        for(int i=0;i<a.length;i++){
             a[i]=rand.nextInt(20); //随机产生 0-20 之间数的赋给数组 a
             System.out.println("a["+i+"]="+a[i]); //打印数组 a
        }
        for(int i=0;i<b.length;i++){
             b[i]=rand.nextDouble();//随机产 double 数的赋给数组 b
             System.out.println("b["+i+"]="+b[i]);//打印数组 b
        }
  }
}
```

将上述代码进行编译并运行，得到如图 4-4 所示的效果。

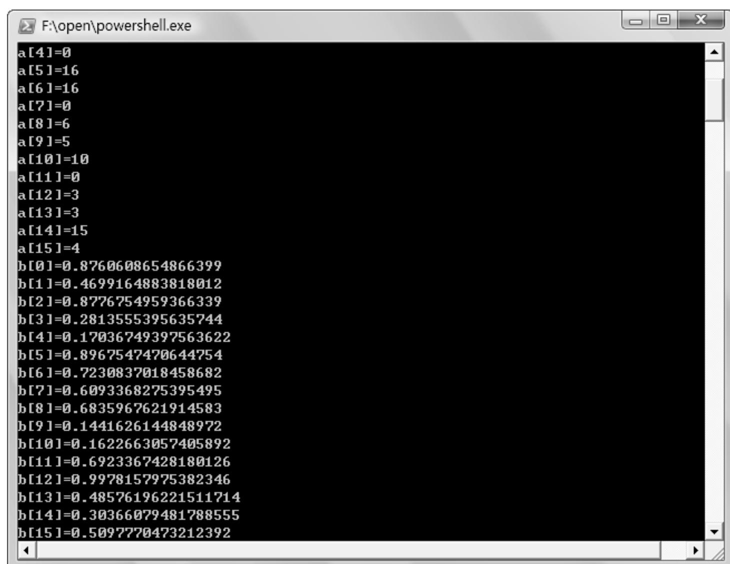


图 4-4　数组的初始化



多学一招

假设想要实现一个功能：产生一组随机数，然后定义数组并将其打印输出，其代码（光盘：源代码/第 4 章/ shunew2.java）如下：

```
import java.lang.Math;
class Array{
   public Array(int i){
         System.out.println("int 型数组"+i);
   }
   public Array(double i){
         System.out.println("double 型数组"+i);
   }
   public Array(int i,int j){
         System.out.println("相乘数组"+i*j);
   }
}
public class ShuNew2{
   public static void main(String[] args) {
         Array[] ArrayB=new Array[5];
         int [] ArrayC=new int[5];
         double[] ArrayD=new double[5];
         for(int i=0;i<ArrayC.length;i++){
               ArrayC[i]=i;
               ArrayB[i]=new Array(ArrayC[i]); }
         for(int i=0;i<ArrayD.length;i++){
               ArrayD[i]=Math.atan(i);
```

```
                ArrayB[i]=new Array(ArrayD[i]);}
            for(int i=0,j=0;i<ArrayC.length;i++,j++){
                ArrayB[i]=new Array(ArrayC[i],ArrayC[j]);
            }     }}
```

将上述代码进行编译并运行，得到如图 4-5 所示的效果。

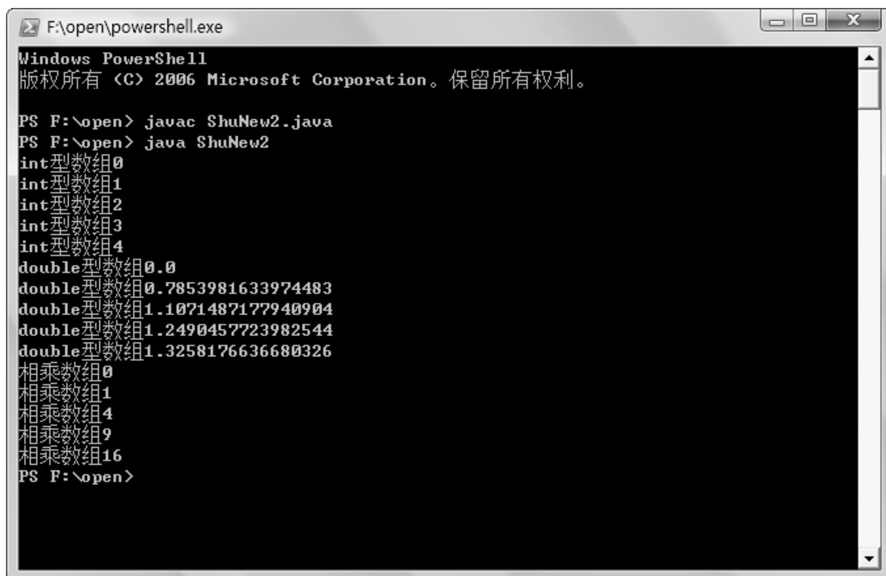


图 4-5　数组初始化

4.2　二维数组

在 Java Web 编程应用中，需要经常使用 Java 二维数组。在二维数组中有两个坐标，第一坐标表示所在行，第二坐标表示所在的列。下过围棋的读者都应该知道棋盘中的行和列，通过行和列就可以确定一个交叉的点，初学者可以将二维数组理解为围棋棋盘。在本节的内容中，将对 Java 二维数组的基本知识进行详细讲解。

4.2.1　二维数组的声明

二维数组是一个特殊的一维数组，其每一个元素又是一个数组。声明二维数组的方法非常简单，具体格式如下。

```
float A[][];
char B[][];
```

参数介绍如下。

- float：数组类型。
- A：数组名称。
- char：数组类型。

- B：数组名称。
- 数组 A 的元素可以存放在 float 型数据。

4.2.2　二维数组的创建

创建二维数组的实质是在电脑上申请存储空间，例如，下面的代码创建了一个二维数组 A。

```
int A[][]={1,3,5,7},{2,4,6,8};
```

在上面的代码中，A 是二维数组的数组名，实质上二维数组相当于一个两行四列的矩阵，当需要取多维中的值时，可以使用下标来显示。其格式如下。

```
Array[i-1][j-1]
```

参数介绍如下。

- i：数组的行数。
- j：数组的列数。

下面以一个二维数组为例，读者看一下 3 行 4 列的数组内部结构，如表 4-1 所示。

表 4-1　二维数组内部结构表

	列 1	列 2	列 3	列 4
行 0	A[0] [0]	A[0] [1]	A[0] [2]	A[0] [3]
行 1	A[1] [0]	A[1] [1]	A[1] [2]	A[1] [3]
行 2	A[2] [0]	A[2] [1]	A[2] [2]	A[2] [3]

通过下面的代码将创建了一个二维数组，并打印输出了二维数组中的数据。其代码（代码 15：光盘：源代码/第 4 章/ shu2.java）如下。

```
public class shu2
{
  public static void main(String[] args)
  {
        int[][] a1={
                    {1,2,3,7},
                    {4,5,6,8},
                    {0,11,12,9},
        };
        for(int i=0;i<a1.length;i++)
              for(int j=0;j<a1[i].length;j++){
                    System.out.println("a1["+i+"]  ["+j+"]="+a1[i][j]);
              }
  }
}
```

将上述代码进行编译并运行，得到如图 4-6 所示的效果。

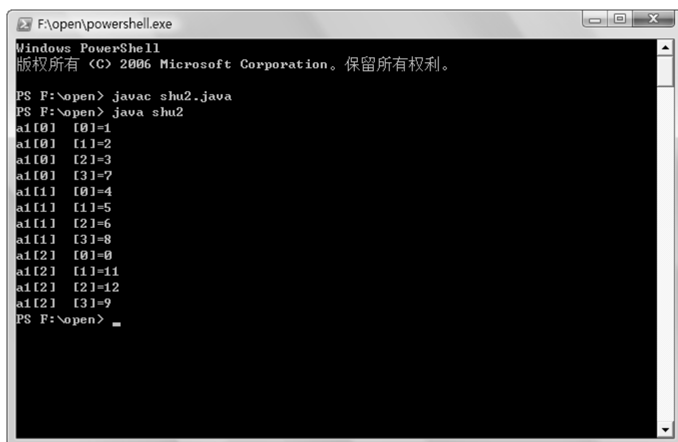


图 4-6　二维数组的创建

4.2.3　初始化二维数组

二维数组的初始化方法和一维数组初始化方法没有什么区别，具体语法格式如下。

```
array=new int[]…[]{第一个元素的值，第二个元素的值，第三个元素的值，…};
```

或使用对象数组的语法格式。

```
array=new int[]…[]{new 构造方法（参数列），{new 构造方法（参数列），…};
```

其参数介绍如下。

- ❑ array：数组名称。
- ❑ new：实例化对象语句。
- ❑ int：数组元素类型。

二维数组是多维数组，必须用多个“{}”括起来。假设在一个二维数组中希望第一维有 3 个索引，第二维有两个索引，就要依下列语法来指定元素的初始值。

```
integer[][]array=new Integer[][]{
{new Integer(1), new Integer(2)},
{new Integer(3), new Integer(4)},
{new Integer(5), new Integer(6)},
}
```

参数介绍如下。

- ❑ array：数组名称。
- ❑ int：数组元素类型。
- ❑ new：实例化对象语句。
- ❑ Integer：数组元素类型。

实例 18：打印一组数组中的最大值和最小值

在本实例中定义了一个二维数组，然后分别打印输出一组数组中的最大值和最小值，其

代码（光盘：源代码/第 4 章/ shuying.java）如下。

```
public class shuying
{
   int grades[][]={
               {77,68,86,73},
               {96,87,89,81},
               {70,90,86,81}
   };
   public static void main(String args[]){
         shuying m=new shuying();
         System.out.println("最小的数："+m.minimum());
         System.out.println("最大的数："+m.maximum());
               }
   public int minimum(){
         int  lowGrade=grades[0][0];
         for(int row=0;row<grades.length;row++){
               for(int column=0;column<grades  [row].length;column++){
                     if(grades[row][column]<lowGrade)
                           lowGrade=grades[row][column];
               }
         }
         return lowGrade;
   }
   public int maximum(){
         int highGrade=grades[0][0];
         for(int row=0;row<grades.length;row++){
               for(int column=0;column<grades  [row].length;column++){
                     if(grades[row][column]>highGrade)
                           highGrade=grades[row][column];
               }
         }
         return highGrade;
   }
}
```

将上述代码进行编译并运行，得到如图 4-7 所示的效果。

```
F:\open\powershell.exe
Windows PowerShell
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

PS F:\open> javac shuying.java
PS F:\open> java shuying
最小的数：68
最大的数：96
PS F:\open> _
```

图 4-7　找数组最大值和最小值

 多学一招

在上面的二维数组实例中，可以在数组中找到最大值和最小值。读者可以通过下面的代码来加深对二维数组的理解，其代码（光盘：源代码/第 4 章/ shuying2.java）如下。

```
public class shuying2
{
   public void array()
 {
   }
   public static void main(String[] args)
{
   final int Startrate=10;
   final int NRATES=6;
   final int NYEARS=10;
   double [] interestRate=new double[NRATES];
   for(int j =0;j<interestRate.length;j++){
   interestRate[j]=(Startrate+j)/100.0;
   }
   double [][] balances=new double[NYEARS][NRATES];
   for(int j =0;j<balances[0].length;j++){
   balances[0][j]=10000;
   }
   for(int i=1;i<balances.length;i++){
   for(int j=0;j<balances[i].length;j++){
   double oldbalance=balances[i-1][j];
   double interest=oldbalance*interestRate[j];
   balances[i][j]=oldbalance+interest;
   }
   }
   for(int j=0;j<interestRate.length;j++){
   System.out.printf("%9.0f%%",100*interestRate[j]);
   }
   System.out.println();
   for(double [] row:balances){
   for(double b:row){
   System.out.printf("%10.2f", b);
   }
   System.out.println();
   }
   }
}
```

将上述代码进行编译并运行，得到如图 4-8 所示的效果。

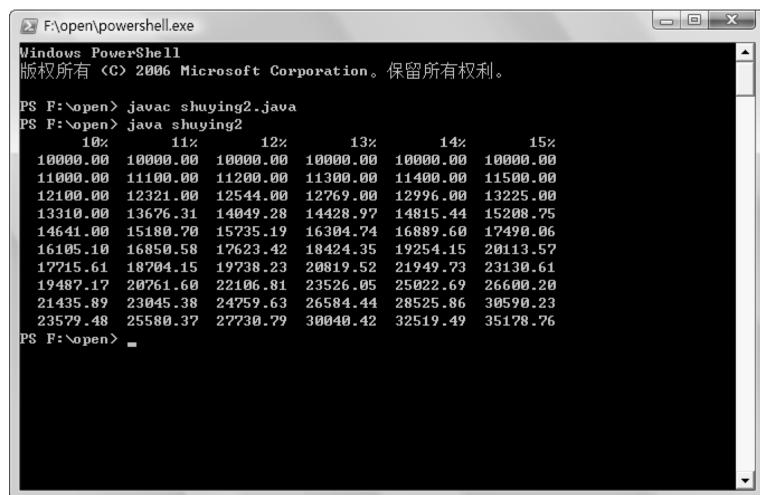


图 4-8　数组的应用

4.3　多维数组

一维数组和二维数组是最常用的数组，除此之外，在 Java 中还有多维数组，例如，三维数组。在多维数组中，三维数组用得最为普遍，在本节的内容中，将主要讲解三维数组的声明和初始化的知识。

4.3.1　声明和创建多维数组

声明多维数组与声明一维数组和二维数组没有什么大的区别，只是多了一个下标，具体格式如下。

```
float a[][][];
char b[][][];
```

参数介绍如下。

- float：数组类型。
- a：数组名称。
- b；数组名称。

掌握了声明多维数组的方法后，创建多维数组就变得非常简单了，例如，下面的代码创建了一个 2×2×3 的三维数组，读者可以将其想象成多个二维数组即可。

```
int[][][] a=new int[2][2][3];
```

4.3.2　初始化多维数组

初始化多维数组的方法十分简单，和初始化一维数组和二维数组没有大的区别。例如，在下面的代码中定义了一个三维数组 a，并进行了初始化。

```
        int[][][]a={   //初始化三维数组
        {{1,2,3}，{4,5,6}}
        {{7,8,9},{10,11,12}}
        }
```

提示： 在 Java 程序中，二维数组应用最为广泛，三维数组也时常被应用，在编写 Java 程序时，一般不建议使用三维以上的数组，因为它会使程序变得难懂，所以在本书中只讲解三维数组。

实例 19：打印三维数组

在本实例中创建一个三维数组，然后打印输出数组中的数据，其代码（光盘：源代码/第 4 章/duoone.java）如下。

```
        public class duoone
        {
           public static void main(String[] args)
           {
                 int[][][]a2=new int[6][3][1];
                 for(int i=0;i<a2.length;i++)
                       for(int j=0;j<a2[i].length;j++)
                             for(int k=0;k<a2[i][j].length;k++)
        {
                                   System.out.println("a2["+i+"]   ["+j+"]["+k+"]
="+a2[i][j][k]);
                             }
           }
        }
```

将上述代码进行编译并运行，得到如图 4-9 所示的效果。

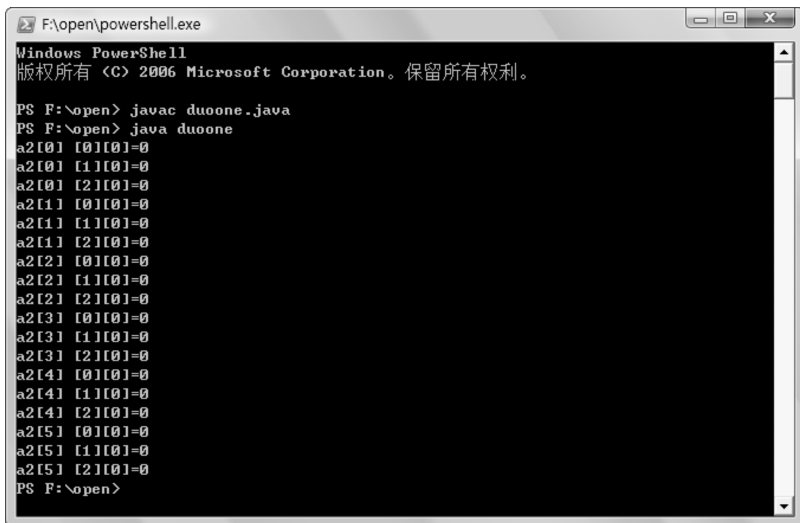
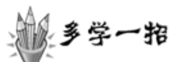


图 4-9　多维数组的打印



在上面的实例中，定义了一个数组，然后定义数组将其打印输出。请读者仔细阅读下面的程序，对多维数组进行更深一步的认识，其代码（光盘：源代码/第 4 章/duoone1.java）如下。

```
public class duoone1
{
   public static void main(String[] args)
   {
         int[][] ar1 = {
                     { 154, 313, 54, 561, 34 },
                     { 54, 444, 123 },
               { 554, 54 }
                     };
         for (int i = 0; i < ar1.length; i++)
         {
         for (int j = 0; j < ar1[i].length; j++)
         System.out.println(ar1[i][j]);
         }
         }
}
```

将上述代码进行编译并运行，得到如图 4-10 所示的效果。

```
F:\open\powershell.exe
Windows PowerShell
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

PS F:\open> javac duoone1.java
PS F:\open> java duoone1
154
313
54
561
34
54
444
123
554
54
PS F:\open>
```

图 4-10　多维数组的简单使用

4.4　操作数组

在 Java Web 中，当使用 Java 语言创建好数组后，通常需要使用 Java 中的方法来控制数组，只有这样才能实现一些具体的功能。在 Java 程序中，当建立好一个数组后，可以对数组进行操作，目的是实现项目中的功能要求，例如，复制数组、排序数组、比较数组和搜索数组等操作。在本节内容中，将一一讲解操作 Java 数组的基本知识。

4.4.1 排序数组

在 Java 程序中，对数组中的元素排序可以使用 sort()方法实现，其格式如下。

```
Arrays.sort(a);
```

参数 a 是排序数组名称。

实例 20：实现数组排序

本实例讲解了数组排序功能的实现过程，其代码（光盘：源代码/第 4 章/Paixu.java）如下。

```
import java.util.Arrays;
public class Paixu
{
        public static void main(String[] args)
        {
             String []s=new String[] {"abcd","ABCE","CEDF","2345"};
             Arrays.sort(s);
             System.out.println(Arrays.asList(s));
        }
   }
```

将上述代码进行编译并运行，得到如图 4-11 所示的效果。

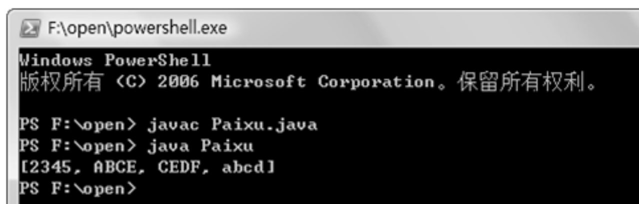


图 4-11 排序数组元素


多学一招

在上面的实例中，对数组的元素进行了排序。下面的代码仍然是实现对数组排序，其代码（光盘：源代码/第 4 章/Paixu1.java）如下。

```
import java.util.Arrays;
public class Paixu1
{
        public static void main(String[] args)
        {
             String []a=new String[] {"569","XYZ","ABC","256"};
             Arrays.sort(a);
             System.out.println(Arrays.asList(a));
        }
```

```
    }
```

将上述代码进行编译并运行，得到如图 4−12 所示的效果。

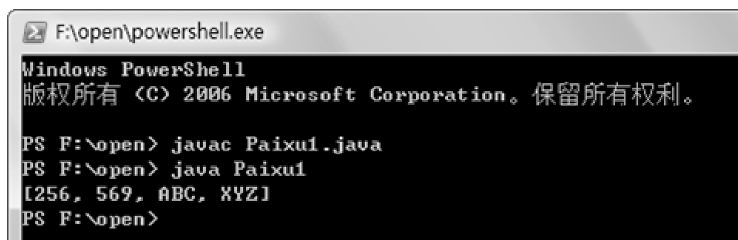


图 4−12　排序数组

4.4.2　对比数组

在 Java 语言中的 Arrays 类中提供了方法 equals()，通过此方法可以对比整个数组是否相同。如果相同则返回 true 值，不相同则返回 false 值。比较数组方法 equals()的格式如下。

```
Arrays.equals(arrayA,arrayB);
```

参数介绍如下。

- arrayA：待比较数组名称。
- arrayB：待比较数组名称。

下面通过一段代码来讲解方法 equals()的用法，其代码（光盘：源代码/第 4 章/duibi.java）如下。

```
import java.util.Arrays;
public class duibi
{
        public static void main(String[] args)
        {
             int[]A1={1,2,3,4,5,6,7,8,9,0};
             int[]A2=new int[9];
             System.out.println(Arrays.equals(A1, A2));
             int[]A3={1,2,3,4,5,6,7,8,9,0};
             System.out.println(Arrays.equals(A1, A3));
             int[]A4={1,2,3,4,5,6,7,8,9,5};
             System.out.println(Arrays.equals(A1, A4));
             System.out.println(Arrays.equals(A2, A3));
             System.out.println(Arrays.equals(A2, A4));
        }
   }
```

将代码进行编译并运行，得到如图 4−13 所示的效果。

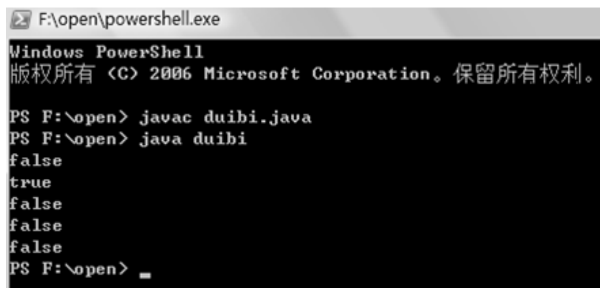


图 4-13　对比数组

4.4.3　复制数组

在 Java 程序中可以自由复制数组，此功能可以通过 System 中提供的特殊方法 arraycopy()来实现，方法 arraycopy()的使用格式如下。

```
System.arraycopy(arrayA,0,arrayB,0,a.length);
```

参数介绍如下。

- array A：来源数组名称。
- 0：来源数组起始位置。
- array B：目的数组名称。
- 0：目的数组起始位置。
- a.length：复制来源数组元素的个数。

上面数组的复制功能有一定局限，其实可以将方法 arraycopy()进行改写，改写后可以复制数组的任何元素，改写后的格式如下。

```
System.arraycopy(arrayA,2,arrayB,3,3);
```

- array A：来源数组名称。
- 2：来源数组起始位置第 2 个元素。
- array B：目的数组名称。
- 3：目的数组起始位置第 3 个元素。
- 3：在来源数组第 2 个元素开始复制 3 个元素。

下面将通过一段代码来讲解复制数组的过程，其代码（光盘：源代码/第 4 章/fuzhi.java）如下。

```
public class fuzhi
{
   public static void main(String[] args)
   {
         int a;
         int Aa[] = { 10,20,30,40,50,60,70,80,90,100 };
         int Bb[] = new int[10];
         System.arraycopy(Aa, 2, Bb, 2, 3);
```

```
            for (a = 0; a < Bb.length; a++)
                System.out.print(Bb[a] + " ");
            System.out.println();
    }
    }
```

将上述代码进行编译并运行，得到如图 4-14 所示的效果。

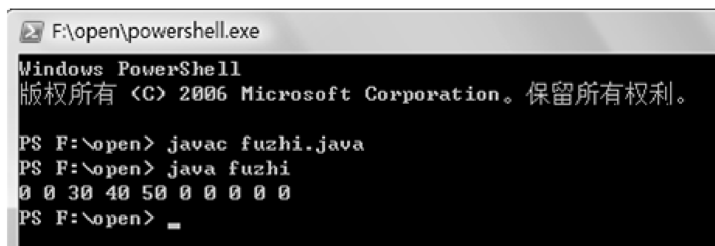


图 4-14　复制数组

4.4.4　填充数组

在 Java 的 Array 类库中，通过填充数组方法 fill()可以填充数组。虽然此方法的功能十分有限，只能用同一个数值填充，但是可以用指定的位置进行填充。方法 fill()填充数组的格式如下。

```
int a []=new int[10];
Arrays.fill(array,11);
```

在上述代码中，a 表示将填充数组的名称，这个程序的含义是将数值 11 填充到数组 a 中。

下面将通过一段代码来讲解填充数组的方法，具体实现代码（光盘：源代码/第 4 章/tianchong.java）如下。

```
import java.util.Arrays;
public class tianchong
{
        public static void main(String[] args) {
            int size=0;
        if(args.length!=0)
            size=Integer.parseInt(args[0]);
        int[]Aa=new int[size];
        Arrays.fill(Aa, 23);
        for(int i=0;i<Aa.length;i++){
            System.out.print("Aa["+i+"]="+Aa[i]+" ");
        }
        System.out.println();
  }
  }
```



注意

通过命令编译并执行上述代码后，不会产生任何结果。读者在学习本书后面的内容后，可以使用 Eclipse 来运行上述代码，在 Eclipse 只需要设置一下运行参数后即可运行本程序。设置方法是依次选择“运行/运行...”命令后打开一个对话框，在里面的具体设置参数如图 4-15 所示。



图 4-15　设置运行参数

设置完成后，即可得到运行效果。

4.4.5　寻找数组

寻找数组实际就是搜索数组，在 Java 中可以使用方法 binarySearch()来查找数组中的数据，具体格式如下。

```
int i=binarySearch（a，”abcde”）;
```

参数介绍如下。

- a：搜索数组的名称。
- abcde：需要在数组中查找的内容。

下面通过一段代码来讲解寻找数组的方法，其代码（光盘：源代码/第 4 章/xunzhao.java）如下：

```
import java.util.Arrays;
import java.util.Comparator;
public class xunzhao
{
        public static void main(String[] args)
        {
             int[] a={11,22,33,44,122,232,3343};
             Arrays.sort(a);
             Arrays.binarySearch(a, 44);
             System.out.print("排序后的数组为：");
```

```
                    for(int i=0;i<a.length;i++){
                         System.out.print(+a[i]+" ");
                    }
                    System.out.println();
                         int location=Arrays.binarySearch(a, 232);
        System.out.println(" 查 找  232  的 位 置 是 "+location+",a["+location+"]
="+a[location]);
                         }
          }
```

将上述代码进行编译并运行，得到如图 4-16 所示的效果。

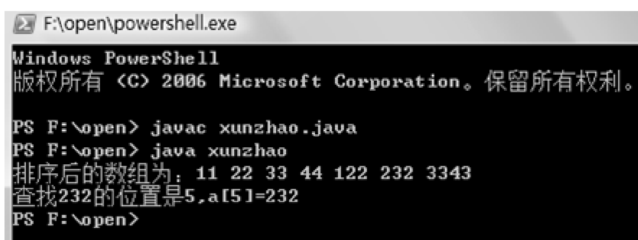


图 4-16　寻找数组

4.4.6　合并数组

有的时候需要将两个数组进行合并后才能操作，究竟如何合并两个数组呢？下面将通过一段代码进行讲解并演示，其代码（光盘：源代码/第 4 章/MergeArray.java）如下。

```
import java.util.*;
public class MergeArray {
public static void main(String[] args) {
  int a[] = {1,2,3};
  int b[] = {2,3,4};
  Set s = new HashSet();
  for(int i=0;i<a.length;i++){
   s.add(a[i]);
  }
  for(int i=0;i<b.length;i++){
   s.add(b[i]);
  }
  Integer c[] = new Integer[5];
  s.toArray();
  for(int i=0;i<s.toArray().length;i++){
   c[i] = (Integer)s.toArray()[i];
   System.out.println(c[i]);
  }
 }}
```

将上述代码进行编译并运行，得到如图 4-17 所示结果。

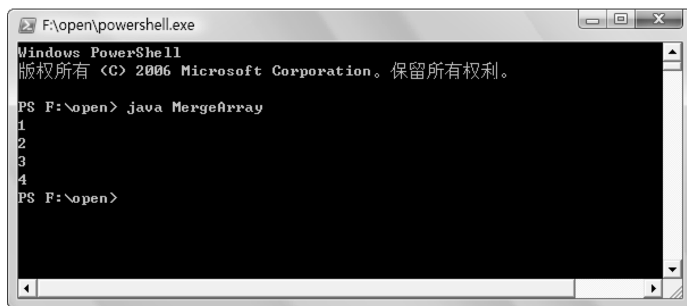


图 4-17　合并数组

4.5　疑难问题解析

在本章的内容中，详细讲解了 Java 语言中数组的基本知识。本节中，将对本章中比较难理解的问题进行讲解。

读者疑问：本章主要讲解了数组，数组的声明、数组的初始化等内容，它们之间有什么异同，对于初学者该如何理解和记忆它？

解答：读者朋友可以对比它们的声明和初始化，它们的初始化和声明十分相似，初学者只要将它们的格式一比较，就会很快理解和记忆它，初学者大可不必死记硬背，很多初学者靠着记忆去学习，这是不对的，应该加以理解，融会贯通，在编写程序时，才可以做到信手拈来。

读者疑问：数组的操作是很重要的，也许数组并不难，难在它的操作上是吗？数组是一种相同数据类型的集合，该怎么样学习与理解数组的操作？

解答：数组的操作只能通过编程多加练习，并没有什么取巧的方法，在本章中，只讲解了 6 个操作数组的方法，实际上，用户用会这 6 个，再加以贯通，可以玩转数组，如可以拆分数组、计算数组元素的总和等。



职场点拨——学编程的误区

一旦选择了程序员这一行业，在学习初期必须耐得住寂寞，收拢住自己的心。学编程是枯燥的，繁琐代码的编写和反复的调试会时刻考验我们的神经。但是学编程也是充满成就感的，每一次调试成功的喜悦会冲掉一切不愉快。既然学编程这么辛苦，我们就要避免一些误区，这样才能提高学习效率。下面是几个学编程的误区。

（1）过分讲究方法和技巧，而不愿意下真功夫

编程是一种技术，也是技能，但这种技能不是专靠技巧能够获得的，而是需要扎实的基础。如果耗费太多的时间和精力去讲究方法和技巧，而忽视对基本知识的学习，反而会影响

学习的效果。只顾钻研技巧，就会在学习内容上花的时间和精力太少，而且养成了投机取巧、不肯下工夫的习惯。方法和技巧只能适当利用，特别适应于有一定开发经验的程序员采用。对于初学者，还是要稳扎稳打，要从自己的学习实践中摸索出适合自己的方法和技巧才会真正管用。

（2）过分讲究速度和效率，不愿花时间经常复习已学过的内容

编程只有靠熟能生巧，练习、练习、再练习，要不断重复才会熟练，只有熟练了才会形成一种不假思索的技能。

（3）没有恒心

编程的熟练要有一个过程，在这个过程中会遇到各种困难，但不能向困难低头，要坚持不懈地反复学习，持之以恒。

（4）忽视界面效果

不重视界面效果，也就是只重技术不重艺术与创造。这个误区在 Web 程序员中比较常见，自以为是后台程序员，所以对页面设计很不屑。其实人们对自然界的感受首先是视觉方面的，如果没有美术基础，只会把项目做得一塌糊涂。

（5）只学而不用

编程的实践性很强，如果只学而不用，就永远也学不好。学编程的目的就是为了应用，要学会在用中学习，这样才能提高兴趣，达到好的学习效果。学编程最忌讳不停地想学新知识，而不扎实地学好基本功，看到新的软件就想学，到最后什么都不精通。

第5章 面向对象

Java 是面向对象的编程语言，面向对象是高级编程语言的精髓思想，面向对象给程序员带来了前所未有的好处，它是程序史上的一次变革。Java 是一门纯面向对象的语言，本章将讲解面向对象的知识。希望大家通过对本章内容的学习，能够对面向对象思想有一个更清楚的认识。通过本章能学到如下知识。

- 类。
- 类的访问。
- 抽象类和抽象方法。
- 类的继承。
- 接口。
- 职场点拨——我们是一个团队。

背景介绍

2008 年 11 月 10 日，多云

刚刚看完一场 NBA 的比赛，由小皇帝詹姆斯领衔的骑士队又输球了，看来有巨星的球队也不一定能换来胜利……



一问一答

小菜：“现在的骑士队有詹姆斯，可是成绩依旧不好。”

Wisdom：“是啊，篮球是 5 个人的运动，单凭詹姆斯一己之力是不能取得好成绩的。”

小菜：“既然篮球场上团队合作很重要，那你说说 Java 体系中最重要的是什么？”

Wisdom：“当然都很重要了，Java 的应用范围太广了，精通任何一个范围都需要花费一定的精力。”

小菜：“本章将要学的面向对象很重要吗？”

Wisdom：“面向对象思想很重要！你一定要了解高级语言的面向对象思想，因为它的推出是软件方式的一大进步，面向对象思想已经成为当前主流语言的核心思想。”

小菜：“面向对象真有这么传奇吗？”

Wisdom：“当然了，几乎所有的高级语言都是基于面向对象的。面向对象在 Java 体系中的地位犹如团体竞技项目中的团队合作。只有明白了面向对象思想，你才能真正认识到 Java 的强大。”

5.1　类

Java Web 的初学者可以这样理解类：就是将相同属性的一些东西放在一起，例如，人、猩猩等，它们都可以被看做一个类。在每一个 Java 源程序中至少包含一个类，类是 Java 面向对象编程思想的基础。接下来将为大家讲解创建与使用类的基本知识。

5.1.1　使用 class 创建类

在面向对象的程序中，有固定特征的都可以定义为一个类。假定熊猫是一个类，则任何一个熊猫都是一个对象。类是一个大概念，所有具有同样属性的东西都可以归纳为一个类。例如，每个熊猫都有眼睛、有鼻子、有嘴……，这就可以把所有熊猫归纳为一个类。当可以肯定一只熊猫有什么样的眼睛、有什么样的鼻子……那么也就可以确定那只熊猫是什么种类的熊猫，这只熊猫也就是熊猫这个类中的一个对象。

在下面的代码中定义了一个熊猫的类 panda，这是具有一定特性的一类事物，而 Tom 则是类的一个对象实例。

```
class panda {
// age 属性
   int age;
// name 属性
   String name;
// shut 方法
   void shut(){
         System.out.println("My name is"+name);
   }
public static void main(String args[]){
//类及类属性和方法的使用
Panda Tom=new Panda();
Tom.age=7;
Tom.name="Tuantuan";
Tom.shut();
}
```

在上面这个例子中，熊猫类 panda 有年龄（age）和姓名（name）两个属性，同时还有 shut()方法。

5.1.2　方法

类的行为都是通过方法来实现的，方法主要由声明和方法体两部分组成，Java 方法的语法格式如下。

```
   [修饰符]<方法返回值类型><方法名>（[参数]）{
方法体
}
```

各个参数的具体说明如下。

- 修饰符：可选参数，用于指定方法的权限，可选值有 public、protected 和 private。
- 方法返回值类型：必选参数，用于指定方法的返回值类型，如果该方法没有返回值，可以使用关键字 void 进行标识，方法返回的数据类型可以是符合 Java 规则的一切数据类型。
- 方法名：必选参数，是程序员指定的方法的名称，方法名必须是合法的 Java 标识符。
- 参数列表：可选参数，是程序员指定的方法所需要的参数。当存在多个参数时，各参数之间应使用逗号隔开，方法的参数类型可以是符合 Java 规则的一切数据类型。
- 方法体：方法体是方法实现部分。

下面通过一段代码来讲解声明方法的过程，具体代码如下。

```
//定义一个无返回值的方法
public void cheng()
{
System.out.println(“我学习 Java Web”);
//…
}
//定义一个有返回值的方法
public int Da()
{
int a=100;
return a;
}
```

在上述段程序里，return 是关键字，功能是将变量 a 的值返回给调用该方法的语句。这就是方法，方法就是用来实现一个功能的元素。

5.1.3　特殊的方法——构造方法

当创建了一个类后，Java 会自动调用该类的构造方法，构造方法的名称必须与类名相同。无论是否定义构造方法，所有的类都有构造方法，如果没有定义，它们将自动调用默认的构造方法。如果定义了构造方法，则就以定义的为准，不会调用默认的构造方法。

构造方法的语法格式如下。

```
[构造方法修饰符]方法名([参数列表])
{
方法体
}
```

下面通过一段代码来讲解构造方法的用法，其代码（光盘：源代码/第 5 章/newbook.java）如下所示。

```
public class newbook
{
   String gname;
```

```
            int    gid;
            float  gprice;
                  public void print()
            {
                  System.out.println("该商品的名字为"+gname+",ID 号"+gid+",价格是
"+gprice);
            }
                  public newbook(String name,int id,float price)
            {
                  gname=name;
                  gid=id;
                  gprice=price;
            }
            public static void main(String args[])
            {
                  newbook book1=new newbook("《原谅我红尘颠倒》",21001,22.0F);
                  book1.print();

                  newbook book2=new newbook("《中国不高兴》",21002,29.8F);
                  book2.print();
                        }
         }
```

将上述代码编译并运行，得到如图 5-1 所示的效果。

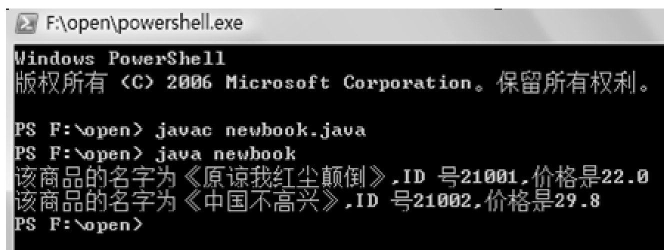


图 5-1　构造方法

5.2　类的访问

在 Java 语言中创建类后，还需要使用 Java 方法来访问创建的类。在创建的类里面主要有方法和属性两种元素，其中属性用于定义类的特征，而方法是类的功能。究竟这些属性和类是如何访问和如何控制的呢？在本节的内容中，将一一进行详细讲解。

5.2.1　修饰符

在一个类中主要有属性和方法两种元素，其中变量是表示类对象的属性，而方法则表示类对象的行为。在定义类的属性和方法时必须指定修饰符，Java 中的访问控制修饰符主要有 public、protected、private 和 final 等，具体说明如下。

- 公共访问控制符 public：用 public 修饰的元素可以被所有其他类引用。由于 public 修饰符会降低运行的安全性和数据的封装性，所以应该尽量少用 public。
- 保护访问控制符 protected：用 protected 修饰的成员变量可以被 3 种类引用。第一种是该类自身；第二种是与它在同一个包中的其他类；第三种是在其他包中的该类的子类。使用修饰符 protected 的主要作用是允许其他包中它的子类来访问父类的特定属性。
- 私有访问控制符 private：用 private 修饰的成员变量（域）只能被该类自身访问，而不能被其他类（包括子类）引用。
- 最终域修饰符 final：是用来定义符号常量的。一个类的域（成员变量）如果被修饰符 final 说明，则它的取值在程序的整个执行过程中都是不变的。
- 默认修饰符：如果没有指定访问控制修饰符，则表示使用默认修饰符，这时变量和方法只能在自己的类及该类同一个包下的类中访问。
- static：被 static 修饰的变量称为静态变量，被 static 修饰的方法称为静态方法。
- transient：它只能修饰非静态的变量。
- volatile：同 transient 一样，它只能修饰变量。
- abstract：被 abstract 修饰的成员称做抽象方法。
- synchronized：只能应用于方法，不能修饰类和变量。

5.2.2　this 的用法

Java 程序中的变量有两种，分别是局部变量和全局变量。读者可以猜想一下，当局部变量和全局变量的名称和类型相同时，这时候全局变量将会被隐藏，不能够使用。为了解决这个问题，Java 中引入了关键字 this，this 关键字的语法格式如下。

```
this.成员变量名
this.成员方法名()
```

下面通过一段代码来讲解 this 的用法，具体代码如下。

```
public class thisone
 {
   public String color="红色";//定义全局变量
   //定义一个方法
   public void hu()
   {
         String color="天蓝色";//定义局部变量
   //下面应用了局部变量
         System.out.print ("她的外套是"+color+"的");
   //下面应用了全局变量
         System.out.print("她的外套是"+this.color+"的");
   }
   }
```

这段代码十分简单，执行后分别输出：

她的外套是天蓝色的

她的外套是红色的

第一个应用的是局部变量，第二个应用的是全局变量。在方法中使用全局变量的时候，加上 this 关键字就可以了。

5.2.3　使用类和对象

使用对象包括引用对象的方法和引用变量两种操作，通过“.”运算符可以实现对变量的访问和对方法的调用。在 Java 程序中，方法和变量都有一定的访问权限，例如，public、protected 和 private 等，通过一定的访问权限来允许或者限制其他对象的访问。

1．创建与使用对象

在编写 Java 程序时，用户可以通过关键字 new 创建对象，电脑会为每一个对象分配内存空间，然后通过该方法访问变量和方法。不同对象的变量也是不同的，方法由对象调用。下面通过一段代码来讲解创建与使用对象的方法，其代码（光盘：源代码/第 5 章/nei.java）如下。

```
public class nei {
   int a;
   int b;
         public void printFoo()
   {
         System.out.println("a="+a+",b="+b);
   }
         public static void main(String args[])
   {
         nei aa=new nei();
         aa.a=1;
         aa.b=2;
         aa.printFoo();

         nei bb=new nei();
         bb.a=4;
         bb.b=9;
         bb.printFoo();
   }
}
```

将上述代码编译并运行，得到如图 5-2 所示的效果。

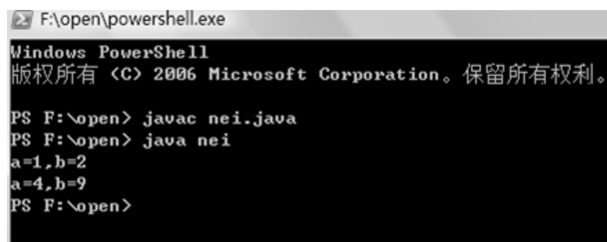


图 5-2　创建和使用对象

2．使用静态修饰符

当声明变量和方法后，如果在前面可以加上 static 关键字修饰符，此时的变量被称做静态变量，此时的方法被称做静态方法。只需要类名即可访问静态变量和静态方法，通过“.”运算符可以实现对变量的访问和对方法的调用。下面通过一段代码来讲解静态修饰符的使用过程，其代码（光盘：源代码/第 5 章/Jingtai.java）如下。

```
public class Jingtai
{
   static int a,b;
   public static int getarea()
{
   return a*b;
}
   public static int getperi()
{
   return 2*(a+b);
}
   public static void main(String args[])
{
   Jingtai a1=new Jingtai();
   Jingtai.a=4;
   Jingtai.b=3;
   System.out.println("area="+Jingtai.getarea());
   System.out.println("peri="+Jingtai.getperi());
}
}
```

将上述代码编译并运行，得到如图 5-3 所示的效果。

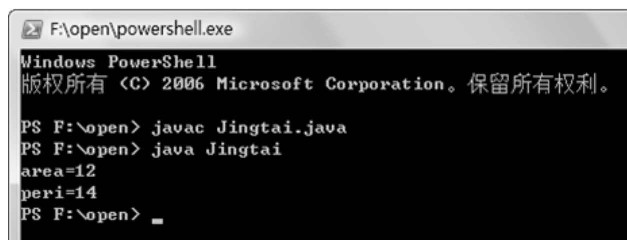


图 5-3　使用静态修饰符

5.3　抽象类和抽象方法

在 Java Web 编程应用中，通常使用 Java 语言编写抽象类和抽象方法来实现具体功能。抽象类是指只声明方法的存在而不去实现它的类，抽象类不能进行实例化，也就是不能创建其对象。在定义抽象类时，要在关键字 class 前面加上关键字 abstract，其语法格式如下。

```
abstract class 类名
{
类体
```

定义了一个抽象类后，那么这个类中至少有一个抽象方法。下面通过一段代码来讲解抽象的规则，其代码（光盘：源代码/第 5 章/Chou.java）如下。

```
abstract class Chou
 {
   int a1;
   int b1;
         Chou(int a,int b)
   {
         a1=a;
         b1=b;
   }
   abstract int mathtext();
}

class Chou1 extends Chou
{
   Chou1(int a,int b)
   {
         super(a,b);
   }

   int mathtext()
   {
         return a1+b1;
   }
}

class Chou2 extends Chou
{
   Chou2(int a,int b)
   {
         super(a,b);
   }
   int mathtext()
   {
         return a1-b1;
   }
}
class Abs3
{
```

```
        public static void main(String args[])
        {
            Chou1 abs1=new Chou1(3,2);
            Chou2 abs2=new Chou2(4,2);
            Chou abs;
            abs=abs1;
            System.out.println("Add is"+abs.mathtext());
            abs=abs2;
            System.out.println("delete is"+abs.mathtext());
        }
    }
```

将上述代码编译并运行，得到如图 5-4 所示的效果。

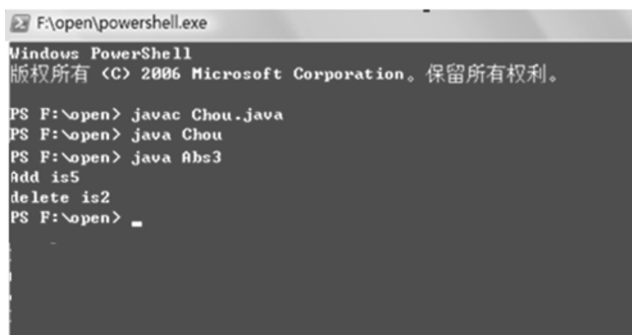


图 5-4　抽象类和方法

5.4　包

在 Java Web 中，直接用 Java 管理包来管理 Java 源文件。当一个 Java 源文件被编译后，都会产生生“class”文件。在一个项目中会产生许多“class”文件，为了方便管理和使用，在 Java 程序引入了包的概念。在本节的内容中，将对包的基本知识进行详细讲解。

5.4.1　定义包

定义包的方法十分简单，只需要在 Java 程序中写一行定义语句就可以了。定义格式如下。

```
package 包名;
```

如果在 Java 源程序中没有 package 语句，定义的所有类都将被包含在一个默认的包中。package 声明了多程序中的类属于哪个包，在一个包中可以包含多个程序。在 Java 程序中还可以创建多层次的包，具体语法格式如下。

```
package 包名1[.包名2[.包名3]];
```

为了加深读者对包的理解，下面通过一段代码来讲解如何定义包，其代码（光盘：源代码/第 5 章/Bao1.java）如下。

```
package china.cq.wz
public class Bao1
{
   int x;
   int y;
   int z;
   void print1(int x)
   {
      System.out.println(2*x);
   }
   void print2(int x,int y)
   {
      System.out.println(2*x+y);
   }
   void print3(int x,int y,int z)
   {
      System.out.println(2*x+y*z);
   }
   void print4(double a)
   {
      System.out.println(2*a);
   }
   void print5(double a,double b)
   {
      System.out.println(2*a+b);
   }
   void print6(double a,double b,double c)
   {
      System.out.println(2*a+b+c);
   }
   public static void main(String[] args)
   {
      Bao1 st=new Bao1();
      st.print1(2);
      st.print2(2,3);
      st.print3(2,3,5);
      st.print4(1.1);
      st.print5(1.1,2.2);
      st.print6(1.1,2.2,3.3);
   }
}
```

将上述代码进行编译并运行，得到如图 5-5 所示的效果。



图 5-5　定义包

5.4.2　导入包

当需要使用系统自带的类、接口或以前编写的包时，需要先导入这些包。导入包的方法十分简单，具体格式如下。

```
import 包名1[.包名2…].(类名1*);
```

参数介绍如下。

- 包名 1：一级包。
- 包名 2：二级包。
- 类名：是需要导入的类的类名。也可使用*号，它表示导入这个包中的所有的类。

下面以一段代码来讲解导入包的方法，其代码（光盘：源代码/第 5 章/GraphTest.java）如下。

```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import javax.swing.BorderFactory;
import javax.swing.ComboBoxModel;
import javax.swing.DefaultComboBoxModel;
import javax.swing.JApplet;
```

```
        import javax.swing.JComboBox;
        import javax.swing.JLabel;
        import javax.swing.JPanel;
        import javax.swing.JSpinner;
        import javax.swing.SpinnerNumberModel;
        import javax.swing.event.ChangeEvent;
        import javax.swing.event.ChangeListener;
        public class GraphTest extends JApplet {
        private static final long serialVersionUID = -3038851324293731224L;
        private PaintingPane paintingPane;
         private Painter currentPainter = Painter.CIRCLE;
         private final static int DEFAULT_RADIUS = 50;
         private int radius = DEFAULT_RADIUS;
         private Color color = getColor();
        public void init() {
          this.setLayout(new BorderLayout());
          paintingPane = new PaintingPane();
          paintingPane.setPreferredSize(new Dimension(250, 250));
          this.add(paintingPane, BorderLayout.CENTER);
           JPanel controlPane = new JPanel();
          final SpinnerNumberModel sm = new SpinnerNumberModel(DEFAULT_RADIUS,
10, 100, 1);
          JSpinner spinner = new JSpinner(sm);
          spinner.addChangeListener(new ChangeListener() {
           public void stateChanged(ChangeEvent e) {
            radius = sm.getNumber().intValue();
            color = getColor();
            repaint();
           }
          });
            final ComboBoxModel cbm = new DefaultComboBoxModel(Painter.values());
          JComboBox combo = new JComboBox(cbm);
          combo.addItemListener(new ItemListener() {
           public void itemStateChanged(ItemEvent e) {
            if (e.getStateChange() == ItemEvent.SELECTED) {
             currentPainter = (Painter) cbm.getSelectedItem();
             color = getColor();
             repaint();
            } }
          });
            JLabel spinnerLabel = new JLabel("Radius: ");
          JLabel comboLabel = new JLabel("Figure: ");
          spinnerLabel.setLabelFor(spinner);
          comboLabel.setLabelFor(combo);
          controlPane.setPreferredSize(new Dimension(150, 250));
          controlPane.setBorder(BorderFactory.createLineBorder(Color.BLACK));
```

```
        controlPane.add(spinnerLabel);
        controlPane.add(spinner);
        controlPane.add(comboLabel);
        controlPane.add(combo);
         this.add(controlPane, BorderLayout.EAST);
        this.setSize(400, 200);
        this.setVisible(true);
       }
      private enum Painter {
        CIRCLE {
         void paint(Graphics g, JPanel panel, int parameter,
            Color color) {
          Dimension d = panel.getSize();
          g.setColor(color);
          g.fillOval(d.width / 2 - parameter, d.height / 2 - parameter,
            parameter * 2, parameter * 2);
         }
         public String toString() { return "Circle"; }
        },
        LINE {
         void paint(Graphics g, JPanel panel, int parameter,
            Color color) {
          Dimension d = panel.getSize();
          int xOffset = d.width / 2, yOffset = d.height / 2;
          g.setColor(color);
          g.drawLine(xOffset - parameter, yOffset, xOffset + parameter,
            yOffset);}
         public String toString() { return "Line"; }
        },
        DIAMOND {
         void paint(Graphics g, JPanel panel, int parameter,
            Color color) {
          Dimension d = panel.getSize();
          int xCenter = d.width / 2, yCenter = d.height / 2;
          int offset = (int) (parameter / SQUARE_ROOT_2);
          int[] xCoors = {
           xCenter - offset, xCenter, xCenter + offset, xCenter,
          };
          int[] yCoors = {
           yCenter, yCenter + offset, yCenter, yCenter - offset,
          };
          g.setColor(color);
          g.fillPolygon(xCoors, yCoors, 4);
         }
         public String toString() { return "Diamond"; }
        },
```

```
TRIANGLE {
 void paint(Graphics g, JPanel panel, int parameter,
    Color color) {
  Dimension d = panel.getSize();
  int xCenter = d.width / 2, yCenter = d.height / 2;
  int offsetPos = (int) (parameter / SQUARE_ROOT_3);
  int offsetNeg = offsetPos / 2;
  int offsetX = parameter / 2;
  int[] xCoors = {
   xCenter - offsetX, xCenter + offsetX, xCenter,
  };
  int[] yCoors = {
   yCenter + offsetNeg, yCenter + offsetNeg,
   yCenter - offsetPos,  };
  g.setColor(color);
  g.fillPolygon(xCoors, yCoors, 3);}
 public String toString() { return "Triangle"; }
},
SQUARE {
 void paint(Graphics g, JPanel panel, int parameter,
    Color color) {
  Dimension d = panel.getSize();
  int xCenter = d.width / 2, yCenter = d.height / 2;
  int offset = parameter / 2;
  int[] xCoors = {
   xCenter - offset, xCenter + offset, xCenter + offset,
   xCenter - offset,
  };
  int[] yCoors = {
   yCenter + offset, yCenter + offset,
   yCenter - offset, yCenter - offset,
  };
  g.setColor(color);
  g.fillPolygon(xCoors, yCoors, 4);}
 public String toString() { return "Square"; }
},
PENTAGON {
 void paint(Graphics g, JPanel panel, int parameter,
    Color color) {
  Dimension d = panel.getSize();
  int xCenter = d.width / 2, yCenter = d.height / 2;
  int xOffset2 = parameter / 2;
  int yOffset3 = (int) (xOffset2 / TANGENT_36);
  int yOffset1 = (int) (xOffset2 / SINE_36);
  int xOffset1 = (int) (yOffset1 * COSINE_18);
  int yOffset2 = (int) (yOffset1 * SINE_18);
```

```
      int[] xCoors = {
       xCenter - xOffset1, xCenter - xOffset2, xCenter + xOffset2,
       xCenter + xOffset1, xCenter,
      };
      int[] yCoors = {
       yCenter - yOffset2, yCenter + yOffset3, yCenter + yOffset3,
       yCenter - yOffset2, yCenter - yOffset1,
      };
      g.setColor(color);
      g.fillPolygon(xCoors, yCoors, 5); }
     public String toString() { return "Pentagon"; }
    },
    HEXAGON {
     void paint(Graphics g, JPanel panel, int parameter,
        Color color) {
      Dimension d = panel.getSize();
      int xCenter = d.width / 2, yCenter = d.height / 2;
      int xOffset = parameter / 2;
      int yOffset = (int) (xOffset * SQUARE_ROOT_3);
      int[] xCoors = {
       xCenter - parameter, xCenter - xOffset, xCenter + xOffset,
       xCenter + parameter, xCenter + xOffset, xCenter - xOffset,
      };
      int[] yCoors = {
       yCenter, yCenter + yOffset, yCenter + yOffset,
       yCenter, yCenter - yOffset, yCenter - yOffset,
      };
      g.setColor(color);
      g.fillPolygon(xCoors, yCoors, 6);  }
     public String toString() { return "Hexagon"; }
    };
    abstract void paint(Graphics g, JPanel panel, int parameter,
         Color color);
    static final double SQUARE_ROOT_2 = Math.pow(2.0, 0.5);
    static final double SQUARE_ROOT_3 = Math.pow(3.0, 0.5);
    static final double SINE_36 = Math.sin(Math.PI / 5);
    static final double TANGENT_36 = Math.tan(Math.PI / 5);
    static final double COSINE_18 = Math.cos(Math.PI / 10);
    static final double SINE_18 = Math.sin(Math.PI / 10); }
  private class PaintingPane extends JPanel {
    private static final long serialVersionUID = 8856506825301606303L;
    public void paint(Graphics g) {
     super.paint(g);
     currentPainter.paint(g, this, radius, color);
    } }
  private static Color getColor() {
    return possibleColors[(int) (Math.random() * 7)]; }
  private static final Color[] possibleColors = {
```

```
        Color.RED, Color.ORANGE, Color.YELLOW, Color.GREEN,
        Color.CYAN, Color.BLUE, Color.MAGENTA,
       };}
```

将上述代码编译并运行，到如图 5-6 所示的效果。

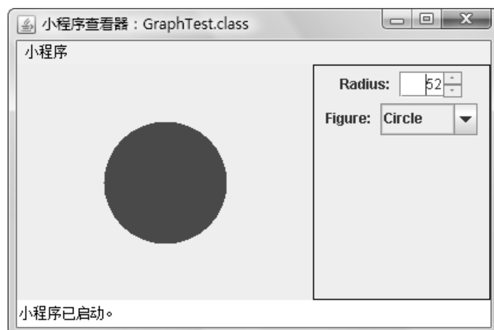


图 5-6　导入包

5.5　类的继承

在 Java Web 中，经常使用 Java 继承类来实现一些功能，这是 Java 程序设计面向对象的一个重要标志。实际上在上一节的学习中已经接触过类的继承，当插入包后，编写的程序一般都有类的继承。究竟类的继承是如何实现的呢？它又有哪些特征呢？在本节将对上述问题进行详细讲解。

5.5.1　继承

继承是指在一个已经定义的类中派生出另外一个类的过程。继承是面向对象的基本机制，利用继承可以创建一个公共类，这个类具有多个项目的共同属性，然后一些具体的类继承该类，同时再加上自己特有的属性。继承的方法十分简单，其语法格式如下。

```
<修饰符>class<子类名>extends<父类名>
{
[<成员变量定义>]…
[<方法的定义>]…
}
```

在通常情况下，人们常把子类称做父类的直接子类，把父类称为子类的直接超类，假如类 A 继承了类 B 的子类，则必须符合下面的要求。

- 类 A 是类 B 的直接子类。
- 存在另一类 C，类 C 是类 B 的子类，类 A 是类 C 的子类，那么可以判断出类 A 是类 B 的子类。

提示：在 Java 程序中，一个类只能有一个父类，即 extends 关键字前只能有一个类，它不支持多重继承。

实例 21：实现类的继承

本实例的目的是讲解类的继承，其代码（光盘：源代码/第 5 章/LeiPai.java）如下。

```
class super1a
{
   String name="玲玲";
   int age=28;
   long number=6666;
}

class super1b extends super1a
{
   String name;
   int age;
   long number;
   void a0()
   {
         name="张爱爱";
         age=20;
         number=88232818;
         name=super.name;
         age=super.age;
         System.out.println("姓名:  "+name);
         System.out.println("年龄:  "+age);
         System.out.println("电话:  "+number);
   }
}

public class LeiPai
{
   public static void main(String args[])
   {
         super1b abc=new super1b();
         abc.a0();

   }
}
```

将上述代码编译并运行，得到如图 5-7 所示的效果。

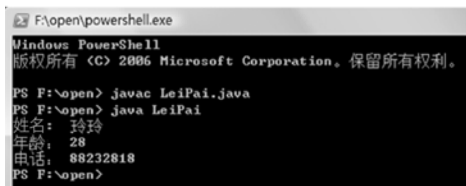
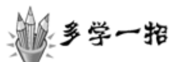


图 5-7　类的继承



类的继承实际上十分简单，在使用类的继承时一定要明白，我们的目的是为了在原来的基础上派生出一个新类，新的类里不但有原来的属性，而且也有新的属性和方法。

5.5.2　对象的使用

假设有类 A 和类 B 两个类，类 A 是子类，类 B 是超类。C 类是 A 类的一个引用，它不但可以指向类 A 的对象，而且也可以指向类 B 的对象。当 C 类指向 B 类时，它可以调用类 B 从类 A 中继承的方法和变量，但是不能调用 B 类新增的变量和方法。为了说明上述问题，下面通过一段代码来讲解使用对象的过程，其代码（光盘：源代码/第 5 章/Duixiang.java）如下。

```
public class Duixiang{
   String  ename;
   int     eid;
   int     eage;
   public String eadder;
         void print(){
         System.out.println("姓名："+ename+" ID 号："+eid+" 年龄："+eage);
   }}
class Duixiang1 extends Duixiang{
   String eadder;
   void print1()
   {
         System.out.println(" 住址："+eaddre);
   }}
class text{
   public static void main(String args[]){
         Duixiang a1=new Duixiang1();
         Duixiang a2=new Duixiang1();
         a1.ename="王二";
         a1.eid=21024;
         a1.eage=55;
         System.out.println("这是一个父类的引用指向子类的对象");
         a1.print();
         System.out.println("这是子类特有的属性");
         a2.eadder="华龙大道";
         a2.print();
   }}
```

将上述代码编译并运行，得到如图 5-8 所示的效果。

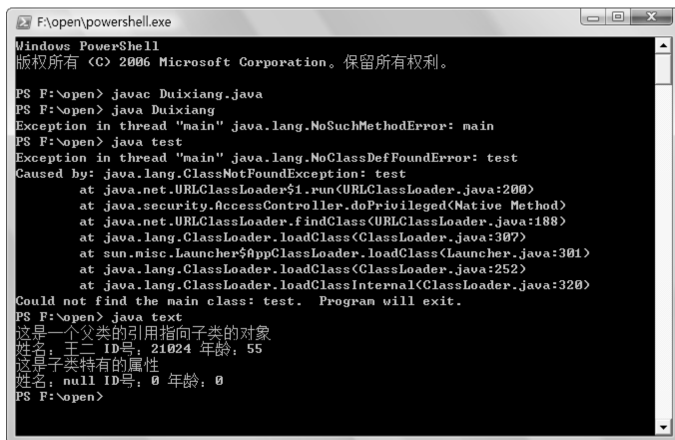


图 5-8　对象的使用

5.5.3　对父类的访问

继承了类之后，如何访问父类的方法、属性和构造方法，是初学者最困惑的问题。因为继承都很简单，按照本书 5.5.1 节介绍的格式就可以实现继承，关键问题是如何访问和利用父类属性和方法。在下面的内容中，将寻找这个问题的答案。

1．调用父类的构造方法

一个类继承另外一个类后，它的子类可以调用父类的构造方法，其格式如下。

```
Super(参数);
```

参数是指父类构造方法所需要的参数，在子类中的 super 必须在第一行执行。当子类明确不使用 super 关键字时，系统会自动调用父类中默认的构造方法。

调用父类的构造方法十分简单，下面通过一段代码进行讲解，其代码（光盘：源代码/第 5 章/ NewBook1.java）如下。

```
public class NewBook1 {
        String bname;
        int    bid;
        int    bprice;
     NewBook1(){
             bname="历史密码";
             bid=32221;
             bprice=32;
        }
        NewBook1(NewBook1 a){
             bname=a.bname;
             bid=a.bid;
             bprice=a.bprice;
        }
   NewBook1(String name,int id,int price){
```

```
                bname=name;
                bid=id;
                bprice=price;
                     }
                     void print(){
                System.out.println("书名："+bname+"序号："+bid+"   价格：
"+bprice);
          }
     }
     class Badder extends NewBook1{
          String badder;
          Badder(){
                super();//调用父类的构造方法
                badder="新世界出版社";
          }
          Badder( Badder b){
                super(b);//调用父类的构造方法
                badder=b.badder;
          }
                     Badder(String x,int y,int z,String aa){
                super(x,y,z);//调用父类的构造方法
                badder=aa;
          }
     }
     class text{
          public static void main(String args[]){
                Badder a1=new Badder();
                Badder a2=new Badder("市委常 委",21006,26,"中国工人出版社");
                Badder a3=new Badder(a2);
              System.out.println(a1.badder);
                a1.print();
                System.out.println(a2.badder);
                a2.print();
                a3.print();
          }
     }
```

将上述代码编译并运行，得到如图 5-9 所示的效果。

```
F:\open\powershell.exe
Windows PowerShell
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

PS F:\open> javac Newbook1.java
PS F:\open> java text
新世界出版社
书名：历史密码序号：32221  价格：32
中国工人出版社
书名：市委常 委序号：21006  价格：26
书名：市委常 委序号：21006  价格：26
PS F:\open> _
```

图 5-9　调用父类的构造方法

2．调用父类的属性和方法

调用父类的属性和方法也是通过 super 关键字实现的，其语法格式如下。

```
Super.[方法和全局变量];
```

调用父类的方法和属性十分简单，下面通过一段代码进行讲解，其代码（光盘：源代码/第 5 章/NeiOne.java）如下。

```
class NeiOne{
   int a;
   int b;
}
class NeiOne1 extends NeiOne{
   int a;
   int b;
   NeiOne1(int x,int y,int z,int q){
         super.a=x;//调用父类被子类隐藏的变量
         super.b=y;
       a=z;
       b=q;
   }
   void print(){
         System.out.println(""+super.a);
         System.out.println(""+super.b);
         System.out.println(""+a);
         System.out.println(""+b);
         }
}
class Nei1{
   public static void main(String args[]){
         NeiOne1 a1=new NeiOne1(1,2,3,4);
         a1.print();
   }
}
```

将上述代码编译并运行，得到如图 5-10 所示的效果。

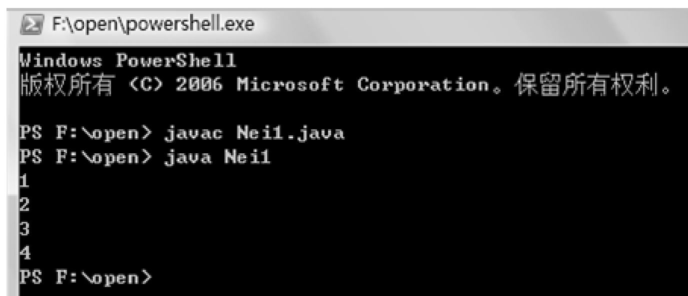


图 5-10　调用类的方法和属性

5.5.4　重载和重写

在面向对象的编程中，常常用重载和重写去修改父类的内容。重载和重写虽然只有一个字的差别，但是操作却是大不相同。

1．重写

在 Java 程序中，当一个子类继承父类后，这个子类会具备父类的所有特征。当子类需要修改父类中的一些方法进行扩展、增大功能时，程序设计者常把这种操作方法称为重写。重写体现了 Java 的优越性，重写是建立在继承关系上的，它使语言结构更加丰富。下面将通过一段代码来讲解重写是怎么一回事，其代码如下。

```
public class Chongxie
{
   void print()
   {
         System.out.println("我是父类的方法");
   }
}
class Chongxieone extends Chongxie
{
   void print()
   {
         System.out.println("我是子类，重写了父类的方法");
   }
}
```

在设计程序的时候，避免不了父类、子类全部重写的情形。在定义一个新类后，必然会有新的特征，不然这个类也没有意义。下面将通过一段代码来讲解父类和子类全部重写的情形，其代码（光盘：源代码/第 5 章/Chong.java）如下。

```
class Gongsi
{
   String sname;
   int    sid;
   int    snumber;
         void print()
{
         System.out.println("公司名："+sname+"  序号："+sid+"公司人数："+snumber);
}
         Gongsi( String name,int id,int number){
         sname=name;
         sid=id;
         snumber=number;
   }
}
```

```
        class Gongsi1 extends Gongsi
        {
           String saddre;
           Gongsi1(String x,int y,int z,String aa)
        {
                super(x,y,z);
                sadder=aa; }
                void print(){
                System.out.println("公司全名为："+sname+"  序号："+sid+"  总人数：
"+snumber+"  公司地址："+sadder);
           }
        }
        class Chong
        {
           public static void main(String args[])
           {
        Gongsi1 a1=new Gongsi1("重庆有为科技有限公司",872221,71,"重庆九龙坡区");
                a1.print();

           }
        }
```

将上述代码编译并运行，得到如图 5-11 所示的效果。

```
Windows PowerShell
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

PS F:\open> javac Chong.java
PS F:\open> java Chong
公司全名为：重庆有为科技有限公司  序号：872221  总人数：71  公司地址：重庆九龙坡
区
PS F:\open> _
```

图 5-11　重写

2．重载

如果在一个类中的方法名相同而参数不同，则称方法的重载。下面通过一段代码来讲解 Java 重载的用法，其代码（光盘：源代码/第 5 章/Chong1.java）如下。

```
        public class Chong1
        {
           String ename;
           int   age;
                void print()
        {
                System.out.println("姓名为："+ename+" 年龄："+age);
           }
                void print(String  a,int b)
           {
                System.out.println("姓名为："+a+" 年龄："+b);
```

```
    }
        void print(String a,int b,int c)
    {
        System.out.println("姓名为："+a+" 年龄："+b+" ID 号："+c);
    }
        void print (String a,int b,double c)
    {
        System.out.println("姓名为："+a+" 年龄："+b+" ID 号："+c);
    }
    }
class text
{
    public static void main(String args[])
    {
        Chong1 a1=new Chong1();
        a1.ename="谭大刚";
        a1.age=21;
        a1.print();
        a1.print("苏菲菲",27);
        a1.print("薛亚飞",16,10025);
        a1.print("于莎莎",25,4125);
    }
}
```

将上述代码编译并运行，得到如图 5-12 所示的效果。

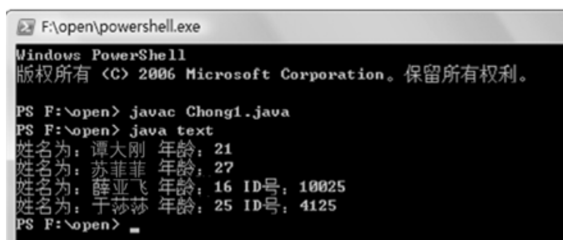


图 5-12　方法的重载

5.6　接口

在 Java Web 项目中，通常使用 Java 接口来实现一些特有的功能。接口实际上是特殊的抽象类，读者已经对抽象类中的抽象方法有了一定的了解，抽象方法只提供函数原型，而没有具体实现的功能。在本节将讲解如何定义接口和使用接口的知识。

5.6.1　接口的定义

创建一个接口后，这个接口的方法也和抽象类中的一样，它的方法是抽象的。也就是说接口不具有实现的功能，它不管指定要做什么和具体怎么做。一旦定义了接口，任何类都可

以实现这个接口，它与类不同，一个类只可以继承一个类，但是一个类可以实现多个接口。在编写程序时，接口解决了一个类要具备多个方面的特征。在 Java 中创建接口的方法十分简单，具体格式如下。

```
[public] interface<接口名>{
[<常量>]
[<抽象方法>]
}
```

参数介绍如下。

- public：接口的修饰符只能是 public，因为只有这样接口才能被任何包中的接口或类访问。
- interface：接口的关键字。
- 接口名：接口的名字，定义法则和类名一样。
- 常量：在接口中不能声明变量，因为接口要具备 3 个特征，分别是 public、static 和 final。

一个类可以实现多个接口，但是实现多接口的类必须符合 3 个原则，第一是能为所有的接口提供实现的功能；第二是能遵循重写的所有规则；第三是能保持相同的返回数据类型。实现接口的方法十分简单，其语法格式如下。

```
[<修饰符>] class<类名> implements <接口名>
{
……
}
```

5.6.2 接口常量和方法

在接口中的量只能是常量，它具备 public、static 和 final 等属性。通过这个特征，保证实现该接口的所有类都可以访问相同的常量。下面将通过一段代码来讲解接口常量的用法，其代码（光盘：源代码/第 5 章/jiekou.java）如下。

```
public interface Jiekou
{
   int a=11;
   int b=21;
   int c=23;
   int d=24;
   int f=25;
         void print();
   void print1();
}
class Jiekou1 implements Jiekou
{
   public void print()
   {
```

```
            System.out.println(a+b);
        }

        public void print1()
        {
            System.out.println(c+d+f);
        }
    }
    class textone
    {
        public static void main(String args[])
        {
            Jiekou1 a1=new Jiekou1();
            a1.print();
            a1.print1();
        }
    }
```

将上述代码编译并运行，得到如图 5-13 所示的效果。

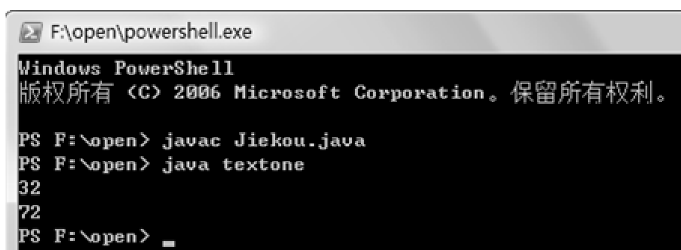


图 5-13　接口

5.6.3　多个接口的实现

在本书前面讲解了单个接口的实现，而在前面也讲解了在 Java 程序中可以实现多个接口。下面将通过一段代码来讲解实现多个接口的方法，其代码（光盘：源代码/第 5 章/jiekou1.java）如下。

```
    interface JieOne{
        int add(int a,int b);}
    interface JieTwo{
        int sub(int a,int b);}
    interface JieThree{
        int mul(int a,int b);}
    interface JieFour{
        int umul(int a,int b);}
    class JieDuo implements JieOne,JieTwo,JieThree,JieFour{
        public int add(int a,int b){
            return a+b;     }
```

```
        public int sub(int a,int b){
            return a-b;     }
        public int mul(int a,int b){
            return a*b;}
            public int umul(int a,int b){
            return a/b;
        }}
    class text{
        public static void main(String args[]){
            JieDuo aa=new JieDuo();
            System.out.println("a+b="+aa.add(20,100));//提供具体实现方法
            System.out.println("a-b="+aa.sub(20,100)); //提供具体实现方法
            System.out.println("a*b="+aa.mul(20,100)); //提供具体实现方法
            System.out.println("a/b="+aa.umul(20,100)); //提供具体实现方法
        }
    }
```

将上述代码编译并运行，得到如图 5-14 所示的效果。

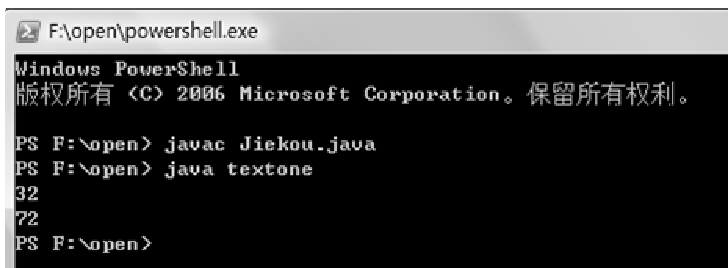


图 5-14　多个接口的实现

5.7　疑难问题解析

在本章的内容中，详细讲解了 Java 面向对象编程思想的基本知识。本节中，将对本章中比较难以理解的问题进行讲解。

读者疑问：听学习过 Java 的前辈讲，Java 有一个 API 手册，可以寻找实现许多功能的接口和类，是吗？该如何去寻找它？

解答：是的，的确有 Java API 手册，它提供了大量的类和接口，以及里面的方法，用户可以去官方网站(http://www.orecle.com/tech net work/java/javase/downloads/index.html)下载获得它。

读者疑问：在网络上搜集 Java 的资料时，看到一个 Java 的内部类，请问什么是 Java 的内部类？

解答：内部类也是一个类，它是类的一个成员，内部类初学者可以把它理解为类的嵌套，一个类被嵌套定义在另一类中，那么这个类称为内部类，对于外部类而言，内部类相当

于变量或和方法，是外部类的一个成员，但是内部类也是属于一个类。



职场点拨——我们是一个团队

现在团队合作出现在很多领域中，例如竞技场上和军事领域。李开复曾经说过：一个人的团队合作能力的重要性甚至超过他的其他能力的重要性！现在有一些程序员在职业发展中遇到问题时，往往怀疑公司不好，待遇不高，怀疑老板和同事……但就是不怀疑自己的团队合作能力。在很多开发公司中，总能看到一些技术一般的程序员做成大事的例子，也总看到一些所谓的技术牛人被埋没的现象。其实成功除了机遇之外，还有一个很重要的因素，那就是程序员的团队合作能力！

那么究竟怎样来衡量程序员团队合作能力呢？建议从以下 5 条规则来衡量。

1）及时关注项目进度和组内情况。

2）以团队利益为思考出发点。

3）团结互助。如果团队中有同事跟不上进度，而你又提前完成了自己的工作，你可以去帮助他。这对你来说虽然带来了额外的工作量，但会增进你们之间的友谊。更重要的是，你为团队按时完成任务提供了保障，这是很有意义的。这种自动“补位”的精神，能显现出一个人的团队合作能力。

4）虚心。

5）友好、宽容地对待新同事。

第6章 异常处理

在使用 Java 语言编写程序时难免会出现异常，异常就是程序在运行时发生的错误或者不正常的情况。在 Java 中提供了完整的异常处理机制，程序出现异常是一件很麻烦的事情，需要程序员来进行检测和处理。但 Java 有一个优点，它可以自动检测异常并对异常进行捕获，并且可以通过程序对异常进行处理。本章将详细讲解如何处理异常，通过本章能学到如下知识：

- 什么是异常处理。
- 捕获异常。
- 抛出异常。
- 异常处理的原则。
- 职场点拨——和上级相处之道。

背景介绍

学校安排小菜在××公司实习。

2009 年 X 月 XX 日，暴雨

今天我很郁闷，项目总监吩咐项目经理和 HR 经理合作完成新项目的策划工作。项目经理老李马上退休了，啥事也不管，他直接将任务给了我。我只是一个程序员，对公司战略和项目规划没有任何经验，让我如何下手啊？



一问一答

小菜：“真烦人啊，这样的领导，啥也不管……”

Wisdom：“呵呵，你对老李的抱怨是很正常的，但是你也要体谅他，即将退休的他虽然看似懒散不求上进了，但是反过来也是给了你机会。具体怎样体谅呢？就需要你认真分析这些老板们的特点了。”

小菜：“也是，我也不能强求他什么了。言归正传，本章异常处理有什么用？”

Wisdom：“在 Java 程序中，错误是不能避免的。为了解决这个问题，Java 推出了异常处理机制。”

6.1 认识异常

通过异常处理能提前解决程序中可能出现的异常，可以通过容错程序使程序继续运行。

究竟什么是异常呢？下面将通过一段代码帮助读者来认识异常。代码（光盘：源代码/第 6 章/kai.java）如下。

```
import java.lang.Math;
public class Subb
{
public static void main(String args[])
{
   Xx Pp=new Xx("Zz");
   Cc Cc=new Cc();
   for(;;)
   {
    try
    {
         Pp.getMyPp(Cc);
    }
    catch(YiChang YiChang)
    {
     System.out.println(YiChang);break;
    }
    Pp.said("处理异常 1.");
    Pp.Aa();break;
   }
}
}
class YiChang extends Throwable
{
public String toString()
{
   return "处理异常 2";
}
}
class Cc
{
private Bb giveABb()
{
   return new Bb();
}
private Dd giveADd()
{
   return new Dd();
}
public Ee giveAEe()
{
   int f=(int)(Math.random()*10);
   if(f>5)
```

```
        return giveABb();
       else
        return giveADd();
    }
    }
    class Bb extends Ee
    {
    Bb()
    {
       name="Gg";
    }
    public void mondoc()
    {
       said("出现异常 3");
    }
    }
    class Dd extends Ee
    {
    Dd()
    {
       name="Ja";
    }
    }

    class Ee
    {
    String name;
    public void said(String words)
    {
       System.out.println(name+":\""+words+"\"");
    }
    }

    class Xx extends Ee
    {
       Xx(String name)
    {
       this.name=name;
    }
    Ee Pp;
    public void getMyPp(Cc Cc) throws YiChang
    {
       Ee Pp=Cc.giveAEe();
       if(Pp.getClass().getName().matches(".*Dd.*"))
        throw new YiChang();
       else
```

```
            this.Pp=(Bb)Pp;
        }
        public void Aa()
        {
            Bb Bb=(Bb)Pp;
            Bb.mondoc();
        }
        }
```

将上述代码进行编译并运行，得到如图 6-1 所示的效果。



图 6-1 处理异常

6.2 什么是异常处理

异常是指 Java 程序中的一些错误，但并不是所有的错误都是异常，并且这些错误有时候是可以避免的。比如代码少了一个分号，运行结果会提示是错误 java.lang.Error；如果在程序中使用了“System.out.println(11/0)”代码，那么是因为用 0 做了除数，所以会抛出“java.lang.ArithmeticException”的异常。

6.2.1 异常处理基础

在编程过程中，首先应当尽可能去避免错误和异常发生。在不可避免、不可预测的情况下，可以考虑在异常发生时如何进行处理。Java 中的异常用对象来表示，Java 对异常的处理是按异常分类处理的，不同的异常有不同的分类，每种异常都对应一个类型（class），每个异常都对应一个异常（类的）对象。

Java 异常有如下两个来源。

1）Java 运行时环境自动抛出系统生成的异常，不管程序员是否愿意捕获和处理，此类异常总要被抛出，比如除数为 0 的异常。

2）程序员自己抛出的异常，这个异常可以是程序员自己定义的，也可以是在 Java 语言中定义的。此类异常用 throw 关键字抛出，这种异常常用来向调用者汇报异常的一些信息。

异常是针对方法来说的，抛出、声明抛出、捕获和处理异常都是在方法中进行的。

在 Java 中通过 5 个关键字来管理异常，分别是 try、catch、throw、throws 和 finally。处理异常的基本过程是：用 try 语句块包住要监视的语句，如果在 try 语句块内出现异常，则会抛出异常，在代码中的 catch 语句块中可以捕获到这个异常并进行处理；还有一部分系统生成的异常在 Java 运行时会自动抛出。另外，也可以通过 throws 关键字在方法中声明该方法要抛出异常，然后在方法内部通过 throw 抛出异常对象。处理异常的 finally 语句块会在方法执行 return 之前执行，一般结构如下。

```
try
{
程序代码
}catch(异常类型 1 异常的变量名 1)
{
程序代码
}catch(异常类型 2 异常的变量名 2)
{
程序代码
}finally
{
程序代码
}
```

Java 异常处理的目的是提高程序的健壮性，功能是在 catch 和 finally 代码块中给程序一个修正机会，让程序不因异常而终止或者流程发生意外而改变。同时，通过获取 Java 异常信息，也为程序的开发维护提供了方便，一般通过异常信息很快就能找到出现异常的问题（代码）所在。Java 异常处理是 Java 语言的一大特色，也是个难点，掌握异常处理可以让写的代码更健壮和易于维护。

6.2.2 异常类

Java 系统里，在 lang 包里有一个名为 Throwable 的类，此类是所有异常的父类或间接父类，每个异常类都是它的子类。其中最常用的子类是 Error 和 Exception，前者用来定义那些通常情况下不希望被捕获的异常，而后者是程序能够捕获的异常情况。表 6-1 中列出了 Java 中常见的异常。

表 6-1 常见的异常

异常类名称	异常类含义
ArithmeticExeption	算术异常类
ArrayIndexOutOfBoundsExeption	数组下标越界异常类
ArrayStroeException	将与数组类型不兼容的值赋值给数组元素时抛出的异常类
ClassCastException	类型强制转换异常类
ClassNotFoundException	未找到相应大类异常
EOFException	文件已结束异常类

（续）

异常类名称	异常类含义
FileNotFoundException	文件未找到异常类
IllegalAccessException	访问某类被拒绝时抛出的异常类
InstantiationException	试图通过 newInstance()方法创建一个抽象类或抽象接口的实例时抛出异常类
IOException	输入/输出抛出异常类
NegativeArraySizeException	建立元素个数为负数的异常类
NullPointerException	空指针异常
NumberFormatException	字符串转换为数字异常类
NoSuchFieldException	字段未找到异常类
NoSuchMethodException	方法未找到异常类
SecurityException	小应用程序执行浏览器的安全设置禁止动作时抛出的异常类
SQLException	操作数据库异常类
StringIndexOutOfBoundsException	字符串索引超出范围异常类

6.3　捕获异常

在 Java Web 开发过程中，作为程序员肯定会设想这个程序在什么时候可能会出现意外和错误，程序员需要通过捕获异常的方法将异常寻找出来，在本节将详细讲解如何捕获 Java 异常的方法。

6.3.1　简单的异常捕获

在 Java 程序中进行异常处理时，最常用的做法是将要处理的异常程序块放在 try 里，然后创建 catch 语句块，下面通过一个例子进行讲解。

实例 22：实现异常捕获和处理

本实例的目的是演示异常的捕获以及简单处理的过程，其代码（光盘：源代码/第 6 章/YiChangone1.java）如下。

```
public class YiChangone1
 {
   public static void main(String args[])
   {
         int a,b;
         try
         {
               a=0;
               b=5/a;
               System.out.println("检验的程序");
         }
         catch(ArithmeticException e)
         {
               System.out.println("经判断，发生 ArithmeticException 异常");
         }
```

```
                System.out.println("程序运行结束");
            }
        }
```

上述程序有一个很明显的错误，那就是 b=5/a，因为除数 a 等于 0，所以发生错误。作为一名程序员，需要将这个错误抛出来并进行处理。将上述代码进行编译并运行，得到如图 6-2 所示的效果。

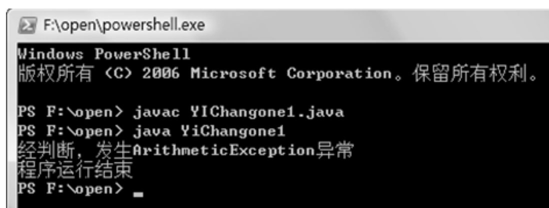


图 6-2　异常的简单处理

多学一招

在上述实例中，讲解了 try…catch 异常处理的用法。再看下面一段代码，目的是使读者对异常处理有一个更深的认识，其代码（光盘：源代码/第 6 章/yichangone2.java）如下。

```
public class YiChangone2
{
    public static void main(String args[])
    {
        int a,b;
        try
        {
            a=0;
            b=5/a;
            System.out.println("需要检验的程序");
        }
        catch(ArithmeticException e)
        {
            System.out.println("异常:"+e);
        }
        System.out.println("结束");
    }
```

将上述代码进行编译并运行，得到如图 6-3 所示的效果。

```
Windows PowerShell
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

PS F:\open> javac YiChangone2.java
PS F:\open> java YiChangone2
异常:java.lang.ArithmeticException: / by zero
结束
PS F:\open>
```

图 6-3　异常处理

6.3.2　多异常捕获

程序员在开发程序的过程中，肯定不会遇到只有一个异常的代码，有时在一个程序中很可能会有许多异常。多个异常该如何捕获呢？例如，在下面代码（光盘：源代码/第 6 章/DuoYiChang.java）中涉及了存在多个异常的情况。

```
public class DuoYiChang
{
   public static void main(String args[])
   {
            int [] a=new int[3];
            try
            {
                 a[3]=123;
                 System.out.println("需要检验的程序");
            }
            catch(ArrayIndexOutOfBoundsException e)
            {
                 System.out.println("发生了 ArrayIndexOutOfBoundsExcep
tion 异常");
            }
            catch(ArithmeticException e)
            {
                 System.out.println("发生了 ArithmeticException 异常");
            }
            catch(Exception e)
            {
                 System.out.println("发生了 Exception 异常");
            }
            System.out.println("结束");
       }
}
```

将上述代码进行编译并运行，得到如图 6-4 所示的效果。

```
Windows PowerShell
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

PS F:\open> javac DuoYiChang.java
PS F:\open> java DuoYiChang
发生了 ArrayIndexOutOfBoundsException 异常
结束
PS F:\open>
```

图 6-4　多异常处理

6.3.3　try…catch…finally 的组合

在使用 try…catch 时可以使用关键字 finally 来处理异常，这个关键字的意思是不管程序

有无异常发生，都将执行 finally 语句块的内容，这使一些不管在任何情况下都必须执行的步骤被执行，从而保证了程序的健壮性。下面通过一段代码来讲解 finally 处理异常的方法，其代码（光盘：源代码/第 6 章/DuoTe.java）如下。

```
public class DuoTe
{
   public static void main(String args[])
{
                        try
            {
                  int age=Integer.parseInt("25L");//抛出异常
                  System.out.println("我要打印 1");
            }
            catch(NumberFormatException e)
            {
                  int b=8/0;
                  System.out.println("年龄请输入整数");
                  System.out.println("错误"+e.getMessage());
            }
            finally {
                  System.out.println("我要打印 2");
                        }
            System.out.println("我要打印 3");
      }
}
```

将上述代码进行编译并运行，得到如图 6-5 所示的效果。



图 6-5　异常的处理

6.4　抛出异常

在 Java 程序中，所谓的抛出异常是指在类中不对异常进行处理，仅将它交给父类来处理。在本节的内容中，将详细讲解抛出异常的基本知识。

6.4.1　使用 throws 抛出异常

抛出异常是指，如果一个方法不处理异常则调用层次向上传递，谁调用这个方法，这个

异常就由谁处理。throws 是抛出异常的一个方法，声明格式如下。

```
Void methodName（int a）throws Exception
{
}
```

实例 23：使用 throws 抛出异常

本实例的目的是讲解使用 throws 抛出异常的方法，其代码（光盘：源代码/第 6 章/Paoyi.java）如下。

```
public class Paoyi
{
   public void methodName(int x) throws
  ArrayIndexOutOfBoundsException,ArithmeticException
   {
         System.out.println(x);
         if(x==0)
         {
               System.out.println("没有异常");
               return;
         }
         else if(x==1)
         {
               int [] a=new int[3];
               a[3]=5;
         }
         else if(x==2)
         {
               int i=0;
               int j=5/i;
         }
   }
   public static void main(String args[])
   {
         Paoyi yc=new Paoyi();
         try
         {
               yc.methodName(0);
         }
         catch(Exception e)
         {
               System.out.println("异常:"+e);
         }
         try
         {
               yc.methodName(1);
         }
```

```
            catch(ArrayIndexOutOfBoundsException e)
            {
                System.out.println("异常:"+e);
            }
            try
            {
                yc.methodName(2);
            }
            catch(ArithmeticException e)
            {
                System.out.println("异常:"+e);
            }
        }
    }
```

将上述代码进行编译并运行，得到如图 6-6 所示的效果。

```
F:\open\powershell.exe
Windows PowerShell
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

PS F:\open> javac Paoyi.java
PS F:\open> java Paoyi
0
没有异常
1
异常:java.lang.ArrayIndexOutOfBoundsException: 3
2
异常:java.lang.ArithmeticException: / by zero
PS F:\open>
```

图 6-6　抛出异常



多学一招

在上述实例中讲解了抛出异常处理的过程，再看下面代码（光盘：源代码/第 6 章/Paoyi/）的实现过程，读者应该加深对异常处理的认识。

先编写文件 MyException.java，其代码如下。

```
public class MyException extends Exception
//自定义异常，这里不要求掌握，在 6.5 节会讲解。
{
  private String name;
  public MyException(String name)
{
        this.name=name;
  }
  public String getMessage()
{
        return this.name;
  }
```

```
    }
```

然后编写文件 Ren.java，其代码如下。

```
public class Ren
{
  public static int check(String strage) throws MyException
{
        int age=Integer.parseInt(strage);
        if(age<0)
        throw new MyException("年龄不能为负数！");
        return age;
  }
  public static void main(String[] args)
{
        try{
             int myage=check("-101");
             System.out.println(myage);
        }catch(NumberFormatException e){
             System.out.println("数据格式错误！");
             System.out.println("原因："+e.getMessage());
        }catch(MyException e){
             System.out.println("数据逻辑错误！");
             System.out.println("原因："+e.getMessage());
        }
  }
}
```

将上述代码进行编译并运行，得到如图 6-7 所示的效果。

```
Windows PowerShell
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

PS F:\open> cd paoyi
PS F:\open\paoyi> javac *.java
PS F:\open\paoyi> java MyException
Exception in thread "main" java.lang.NoSuchMethodError: main
PS F:\open\paoyi> java Ren
数据逻辑错误!
原因：年龄不能为负数!
PS F:\open\paoyi> _
```

图 6-7　抛出异常

6.4.2　使用 throw 抛出异常

在 Java 中除了使用关键字 throws 抛出异常外，还可以使用 throw 关键字来抛出异常，throw 能够把异常抛给上一级调用。抛出的异常可以是异常引用，也可以是异常对象。下面将通过一个例子来讲解使用 throw 抛出异常的方法。

实例 24：使用 throw 抛出异常

本实例的目的是讲解使用 throw 抛出异常的方法，其代码（光盘：源代码/第 6 章/

Yipao1.java）如下。

```
public class Yipao1
{
   void methodName()
   {
         try
         {
               throw new ArrayIndexOutOfBoundsException();
         }
         catch(ArrayIndexOutOfBoundsException aoe)
         {
               throw aoe;
         }
   }
   public static void main(String args[])
   {
         Yipao1 yc=new Yipao1();
         try
         {
               yc.methodName();
         }
         catch(ArrayIndexOutOfBoundsException aoe)
         {
               System.out.println("异常:"+aoe);
         }
   }
}
```

将上述代码进行编译并运行，得到如图 6-8 所示的效果。

```
Windows PowerShell
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

PS F:\open> Javac Yipao1.java
PS F:\open> Java Yipao1
异常:java.lang.ArrayIndexOutOfBoundsException
PS F:\open>
```

图 6-8　使用 throw 抛出异常


多学一招

在上面的实例中讲解了 throw 抛出异常处理，再看下面的一段代码，读者应该加深对异常处理的认识。其代码（光盘：源代码/第 6 章/Paoyi2.java）如下。

```
public class Paoyi2
{
   public static void main(String args[])
```

```
        {
                try
                {
                        throw new ArrayIndexOutOfBoundsException();
                }
                catch(ArrayIndexOutOfBoundsException aoe){
                        System.out.println("异常:"+aoe);
                }
                try
                {
                        throw new ArithmeticException();
                }
                catch(ArithmeticException ae)
                {
                        System.out.println("异常:"+ae);
                }
    }
    }
```

在上述代码中，使用 throw 多次抛出了异常，编译并运行，得到如图 6-9 所示的效果。

```
Windows PowerShell
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

PS F:\open> javac Paoyi2.java
PS F:\open> java Paoyi2
异常:java.lang.ArrayIndexOutOfBoundsException
异常:java.lang.ArithmeticException
PS F:\open>
```

图 6-9　抛出多次异常

6.5　自定义异常的创建

前面讲解了抛出异常的两种方式。在实际编程应用中，只有这两种方法是肯定不够用的。在 Java 中经常需要程序员自己创建异常去满足一些特殊要求，在创建自定义异常时，只需要继承 Throwable 类或者它的子类 Exception 即可。自定义异常让系统把它看成一种异常来对待。由于自定义异常都继承于 Throwable 类，所以也继承了它的方法。在 Throwable 类中包含了如下方法。

- fillInStack Trace()
- getCause()
- getLocalizedMessage()
- getMessage()
- getStack Trace()
- initCause(Throwable cause)

- printStack Trace()
- printStack Trace(PrintStream s)
- printStack Trace(PrintWriter s)
- setStack Trace(Stack TraceElement[] stack Trace)
- toString()

下面通过一段代码来讲解自定义异常的过程，其代码（光盘：源代码/第 6 章/zipao.java）如下。

```
class MyException extends Exception
{
   private int chengji;
   MyException(int a)
   {
         chengji=a;
   }
   public String toString()
   {
         return "异常:"+chengji;
   }
}
public class zipao
{
   void methodName(int a) throws MyException
   {
         System.out.println(a);
         if(a<60)
         {
               throw new MyException(a);
         }
         else
         {
               System.out.println("正常");
         }
   }
   public static void main(String args[])
   {
         Zipao yc=new zipao();
         try
         {
               yc.methodName(75);
               yc.methodName(55);
         }
         catch(MyException e)
         {
               System.out.println(e);
```

```
            }
        }
    }
```

将上述代码进行编译并运行，得到如图 6-10 所示的效果。

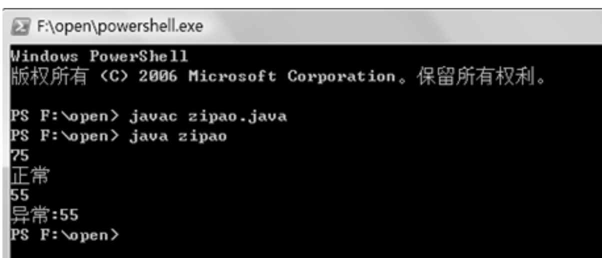


图 6-10　自定义异常

6.6　异常处理的原则

在异常处理的时候并不能控制程序的流程，它的作用仅仅是捕获程序在运行时发生的异常并对它进行处理，保证程序在错误修复后能够继续执行。在编写程序时，如果某个方法发生了异常，则可以采取以下 3 种方式来处理。

1）在当前方法中使用 try…catch 语句捕获并处理异常。

2）在方法声明处，通过 throws 关键字将异常向上抛出。

3）使用 catch 语句捕获异常后，使用 throws 关键字重新抛出异常。



当一个方法有多种异常时，需要使用多个 catch 语句进行处理。

6.7　杜绝异常处理的陋习

有很多初学者，Java 知识学得很好，但是养成了一些陋习，特别是异常处理的陋习。本节将通过一段代码来讲解常见的异常陋习，目的是让读者彻底摆脱容易犯的陋习，写出更规范、更健壮的 Java 代码。

下面这段代码有 6 个问题。

```
OutputStreamWriter out = ...
java.sql.Connection conn = ...
try { // ⑸
 Statement stat = conn.createStatement();
 ResultSet rs = stat.executeQuery(
   "select uid, name from user");
 while (rs.next())
 {
```

```
9    out.println("ID：" + rs.getString("uid") // ⑹
10      "，姓名：" + rs.getString("name"));
11   }
12   conn.close(); // ⑶
13   out.close();
14 }
15 catch(Exception ex) // ⑵
16 {
17   ex.printStackTrace(); //⑴，⑷
18 }
```

作为一名 Java 程序员，至少应该能够找出两个问题。如果不能找出全部 6 个问题，说明在程序的书写上还有许多陋习。下面将一一讲解这几个问题。

6.7.1 丢弃异常

在上述代码的 15～18 行，捕获了异常却没有做任何处理，这就是丢弃异常。如果看到了这种丢弃（而不是抛出）异常的情况，可以断定代码存在问题。当然在极少数情况下，这段代码有存在的理由，但最好加上完整的注释，以免引起别人误解。

这段代码的错误在于异常（几乎）总是意味着某些事情不对劲，或者说至少发生了某些不寻常的事情，不应该对程序发出的求救信号保持沉默和无动于衷。调用 printStackTrace()算不上“处理异常”。不错，调用 printStackTrace()对调试程序有帮助，但程序调试阶段结束之后，printStackTrace()就不应再在异常处理模块中担负主要角色了。丢弃异常的情形非常普遍，当打开 JDK 的 ThreadDeath 类的文档时，可以看到如下说明：

“虽然出现 ThreadDeath 是一种‘正常的情形’，但 ThreadDeath 类是 Error 而不是 Exception 的子类，因为许多应用会捕获所有的 Exception 然后丢弃它不再理睬”。

这段话的意思是，虽然 ThreadDeath 代表的是一种普通的问题，但鉴于许多应用会试图捕获所有异常而不进行适当的处理，所以 JDK 把 ThreadDeath 定义成 Error 的子类，因为 Error 类代表的是一般的应用不应该去捕获的严重问题。由此可见，丢弃异常这一坏习惯很常见，它甚至已经影响到了 Java 本身的设计。那么，应该怎样改正呢？主要有如下 4 个选择。

1）处理异常：针对该异常采取一些行动，例如，修正问题、提醒某个人或进行其他一些处理，要根据具体的情形确定应该采取的动作。再次说明，调用 printStackTrace 算不上已经“处理好了异常”。

2）重新抛出异常：处理异常的代码在分析异常之后，认为自己不能处理它，重新抛出异常也不失为一种选择。

3）把该异常转换成另一种异常：大多数情况下，这是指把一个低级的异常转换成应用级的异常（其含义更容易被用户了解的异常）。

4）不要捕获异常。

提示：既然捕获了异常，就要对它进行适当的处理。捕获异常之后又把它丢弃，不予理睬，这种处理异常的方法直接影响 Java 程序的健壮性。

6.7.2　不指定具体的异常

代码第 15 行虽然交代了异常，但是指定不具体。许多时候人们会被这样一种“美妙的”想法吸引——用一个 catch 语句捕获所有的异常，最常见的情形就是使用 catch(Exception ex)语句。但实际上，在绝大多数情况下，这种做法不值得提倡。要理解这样做的原因，必须完全理解 catch 语句的用途。catch 语句表示预期会出现某种异常，而且希望能够处理该异常。异常类的作用就是告诉 Java 编译器想要处理的是哪一种异常。因为绝大多数异常都直接或间接从 java.lang.Exception 派生，所以 catch(Exception ex)相当于表明想要处理几乎所有的异常，这显然是很不实际的一种做法。

再来看看上面的代码，它真正想要捕获的异常是什么呢？最明显的一个是 SQLException，这是 JDBC 操作中常见的异常，另一个可能的异常是 IOException，因为它要操作 OutputStreamWriter。显然，在同一个 catch 块中处理这两种截然不同的异常是不合适的。如果用两个 catch 块分别捕获 SQLException 和 IOException 就要好多了。这就是说，catch 语句应当尽量指定具体的异常类型，而不应该指定涵盖范围太广的 Exception 类。

另一方面，除了这两个特定的异常外，也可能出现其他许多异常。例如，由于某种原因，executeQuery 返回了 null，可以让它们继续抛出，即不必捕获也不必处理。实际上，我们不能也不应该去捕获可能出现的所有异常，程序的其他地方还有捕获异常的机会，直至最后由 JVM 处理。

提示：在 catch 语句中尽可能指定具体的异常类型，必要时使用多个 catch。不要试图处理所有可能出现的异常。

6.7.3　占用资源不释放

上述代码的 3～14 行就属于占用资源不释放的情况。

异常改变了程序正常的执行流程。这个道理虽然简单，却常常被人们忽视。如果程序用到了文件、Socket、JDBC 连接之类的资源，即使遇到了异常，也要正确释放占用的资源。为此，Java 提供了一个简化这类操作的关键词 finally。

finally 的功能是，不管是否出现了异常，finally 保证在 try…catch…finally 块结束之前，执行清理任务的代码总是有机会执行。但是在编写 finally 块时应当多加小心，特别是要注意在 finally 块之内是否抛出异常，这是执行清理任务的最后机会，尽量不要再出现难以处理的错误。

提示：要想保证所有资源都被正确释放，必须充分运用 finally 关键词。

6.7.4　不说明异常的详细信息

上述代码的 3～18 行，如果循环内部出现了异常，究竟会发生什么事情？我们可以得到足够的信息判断循环内部出错的原因吗？显然不能。我们只能知道当前正在处理的类发生了某种错误，却不能获得任何信息判断导致当前错误的原因。

通过 printStackTrace 的堆栈跟踪功能，能够显示出程序运行到当前类的执行流程，但只提供了一些最基本的信息，未能说明实际导致错误的原因，同时也不易解读。所以在出现异

常时，最好能够提供一些文字信息，例如，当前正在执行的类、方法和其他状态信息，包括以一种更适合阅读的方式整理和组织 printStackTrace 提供的信息。

提示：在异常处理模块中提供适量的错误原因信息，组织错误信息使其易于理解和阅读。

6.7.5 过于庞大的 try 块

在上述代码的 3～14 行，会发现很多程序员经常把大量的代码放入单个 try 块中，这是很不好的习惯。可是很多人为了省事，不愿花时间分析一大块代码中哪几行代码会抛出异常、异常的具体类型是什么，所以会有很多程序员养成这个习惯。如果把大量的语句装入单个巨大的 try 块中，就像出门旅游时把所有日常用品硬塞入一个大箱子，虽然东西是带上了，但要找出来可不容易。一些新手常常把大量的代码放入单个 try 块中，然后在 catch 语句中声明 Exception，而不是分离各个可能出现异常的段落并分别捕获其异常。这种做法为分析程序抛出异常的原因带来了困难，因为一大段代码中有太多的地方可能抛出 Exception。

提示：在异常的处理中，尽量减小 try 块的体积。

6.7.6 输出数据不完整

在上述代码中的 7～11 行，出现了不完整数据的问题。如果循环的中间抛出了异常，则循环的执行是要被打断的，其次，catch 块会被执行吗？已经输出的数据怎么办？结果是使用这些数据的人或设备将收到一份不完整的（因而也是错误的）数据，却得不到任何有关这份数据是否完整的提示。对于有些系统来说，数据不完整可能比系统停止运行带来更大的损失。为解决这个问题，可以向输出设备写一些信息，声明数据的不完整性；另一种解决办法是，先缓冲要输出的数据，准备好全部数据之后再一次性输出。

根据上面的讨论，可以用下面的代码改进，让读者更清楚地认识异常处理的重要性和技巧性，它有了比较完备的异常处理机制，其代码如下。

```
        OutputStreamWriter out = ...
        java.sql.Connection conn = ...
        try {
          Statement stat = conn.createStatement();
          ResultSet rs = stat.executeQuery(
            "select uid, name from user");
          while (rs.next()) {
             out.println("ID：" + rs.getString("uid") + "，姓名：" + rs.
getString("name"));
          } }
        catch(SQLException sqlex) {
          out.println("警告：数据不完整");
          throw new ApplicationException("读取数据时出现 SQL 错误", sqlex);
        }
        catch(IOException ioex) {
          throw new ApplicationException("写入数据时出现 IO 错误", ioex);
        }
        finally
```

```
        {
          if (conn != null) {
            try {
              conn.close();
            }
            catch(SQLException sqlex2) {
              System.err(this.getClass().getName() + ".mymethod - 不能关闭数据库连
接: " + sqlex2.toString());
            }
          }
          if (out != null) {
            try {
              out.close();
            }
            catch(IOException ioex2) {
              System.err(this.getClass().getName() + ".mymethod - 不能关闭输出文件
" + ioex2.toString());
            }   } }
```

6.8　疑难问题解析

本章，详细讲解了 Java 语言中异常处理的基本知识。本节将对本章中比较难以理解的问题进行讲解。

读者疑问：处理异常看上去十分简单，但真正处理起来却十分麻烦。处理异常的能力对于一个程序员有多重要？有没有什么绝招？

解答：处理异常可以间接地衡量一个人的 Java 水平，一个衡量Java设计师水平和开发团队纪律性的好方法就是读一读他们应用程序里的异常处理代码。首先要注意的是有多少代码用于捕获异常，写进日志文件，决定发生了什么和在不同的异常间跳转。干净、简捷、关联性强的异常处理通常表明开发团队有着稳定的使用 Java 异常的方式。

读者疑问：在抛出异常处理时，Java 提供了两种方法，即通过 throw 关键字和 throws 关键字处理，它们有什么不同？

解答：throw 语句用在方法体内，表示抛出异常，由方法体内的语句处理，不能单独使用，要么和 try…catch 一起使用，要么和 throws 一起使用；throws 语句用在方法声明后面，表示这个方法可能会抛出异常，表示的是一种倾向、可能，但不一定实际发生。



职场点拨——和上级相处之道

在职场中和上级相处很重要，因为我们在职场中不可避免地会出现一些错误，而我们的错误大多数是由上级来处理的。所以只要解决好和上级的关系，就能大大减少出错。并且我

们的升职和工作调动，都和上级密切相关。鉴于上级的重要性，告知读者在和上级相处时应该遵循如下两个原则。

（1）察言观色

任何人的脸色都会因为遇到不幸或霉运而改变，上级当然也不会例外。当然也不排除有喜怒不形于色的人，但是这种人毕竟是少数。职场中的大多数上级都已届中年，地位越高责任就越重。如果工作能顺利进行，他们就会有很高的成就感，喜形于色；如果不能，他们也会有很强的挫败感，不快的情绪也溢于言表。所以我们要时刻观察上级的表情，当上级高兴时，我们要努力工作，并拉近和上级的距离；当上级不高兴时，要用实际行动来为上级分忧。

（2）体会暗示

暗示就是用间接含蓄的方式来表述自己的意见和态度。在与上级相处的过程中，不能忽视上级对你的暗示。为了维护上级的尊严和威信，也需要及时、准确地给上级以某种暗示回应。上级说话经常话里有话，下级要学会听弦外音。上级的一些想法、观点和意图，都能通过他的言谈和举止表露出来。作为一名好的下级，可以凭借弦外音就能与上级心有灵犀。能够听出上级的弦外音，并尽可能地满足上级的要求，这样的人才能更好地获得上级信任。

第二篇　提　高　篇

第 7 章　搭建 Java Web 开发环境

经过前面的学习，相信读者已经掌握了 Java 的基础知识。但是通过记事本编写 Java 程序和 Java Web 程序是一件十分不现实的事情，从本章内容开始，将教会读者理解 Java 的运行环境，学会使用 Java 和 Java Web 的开发工具。通过本章能学到如下知识。

- 下载和安装 Tomcat。
- 下载和安装 MyEclipse。
- 职场点拨——一份简历的思考。

背景介绍

2009 年 XX 月 X 日，天气阴。

实习已经一个月了，我还在想着毕业后的工作问题。今天在招聘版上看见一个感兴趣的招聘信息。

职位：AA 公司招聘软件工程师

要求如下：

计算机相关专业，专科以上学历，2 年以上工作经验。

熟练运用 Java 语言，熟悉其他编程语言，例如，VB、C 等优先。

熟练使用 SQL Server 2005 数据库，熟悉 MySQL 者优先。

熟悉 Windows 开发环境，熟悉 Linux 者优先。

工作认真细心，能吃苦耐劳，有上进心，有团队合作精神和良好的沟通能力。

我很感兴趣，决定写一份简历去试一试。



一问一答

小菜：“进行 Java Web 开发还得需要搭建专门的开发环境吗？”

Wisdom：“对，你步入职场的第一步是写简历找工作，作为 Java Web 开发的第一步是搭建开发环境。要运行 Java 文件，首先要安装好 JDK，在前面已经讲解过，然后用户只需要安装一个运行 Java Web 运行的服务器就可以了。”

7.1　下载和安装 Tomcat

Tomcat 是 Java Web 运行的服务器软件，用户要开发和运行 Java Web 程序，就必须下载和安装 Tomcat。在安装 Tomcat 前，一定要先安装和配置 JDK，在本书第 1 章已经讲解了如何安装和配置 JDK，这里不再赘述，本节将讲解如何下载和安装 Tomcat。

7.1.1　下载 Tomcat

Tomcat 是一款免费的服务器软件，用户可以在它的官方网站下载各种版本，这里就以下载 Tomcat 最新版本为例进行讲解，其具体操作如下。

1）打开浏览器，在地址栏中输入“http://tomcat.apache.org/”，进行浏览，如图 7-1 所示。

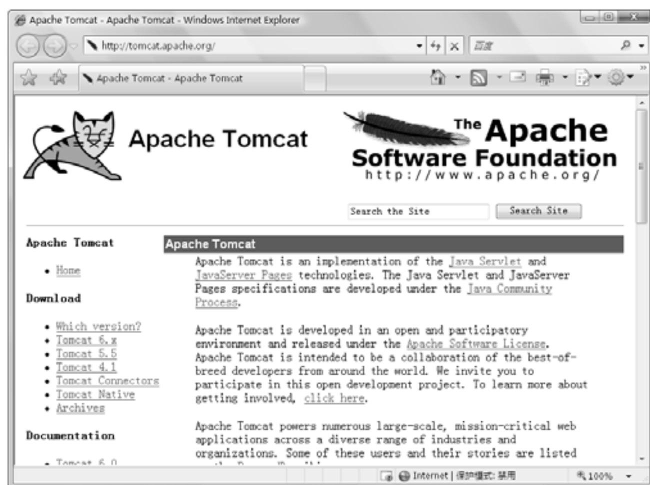


图 7-1　Tomcat 的首页

2）单击左边的“Tomcat 6.x”超链接，在打开的新页面中，将网页拉到最下面，如图 7-2 所示。

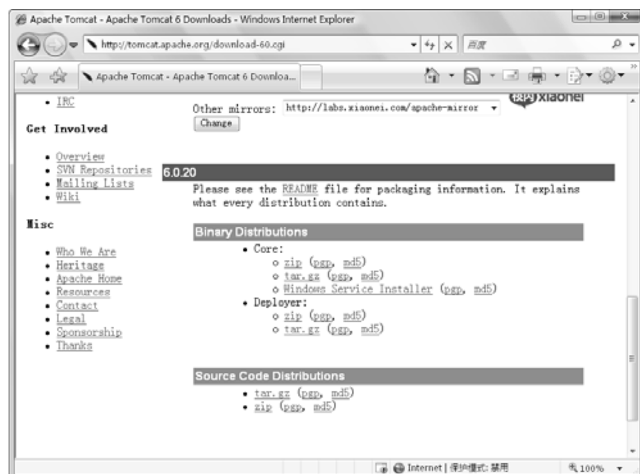


图 7-2　下载最新版本

3）单击 “Windows Service Installer (pgp, md5)” 超级链接，会打开下载窗口，如果系统安装了迅雷下载工具，将会打开如图 7-3 所示界面。

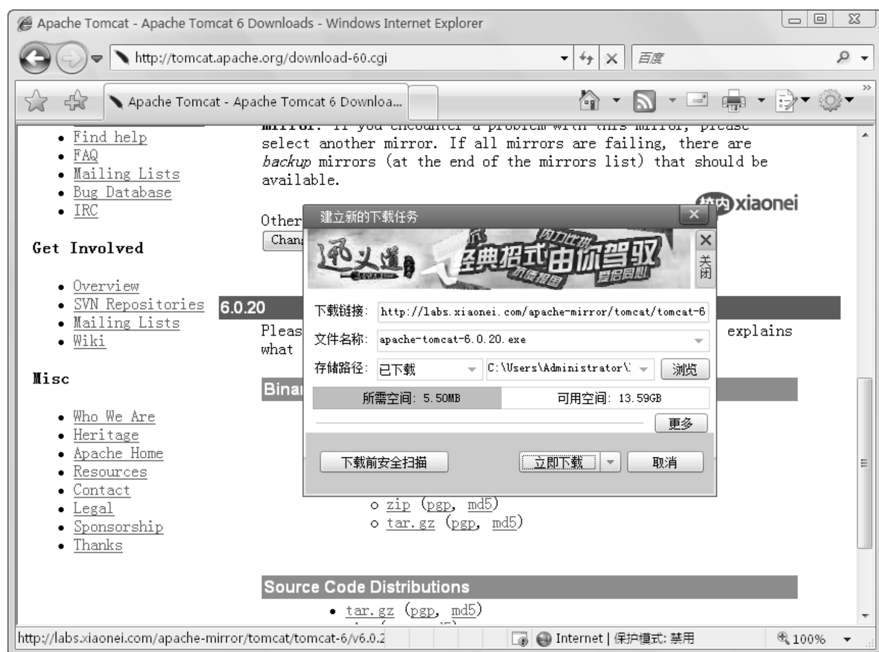


图 7-3　下载 Tomcat

7.1.2　安装 Tomcat

安装 Tomcat 服务器软件十分简单，只要双击下载的安装文件，按照向导提示操作即可实现，其具体操作过程如下。

1）双击下载的 Tomcat 软件，即可对它进行安装，单击“next”按钮，如图 7-4 所示。

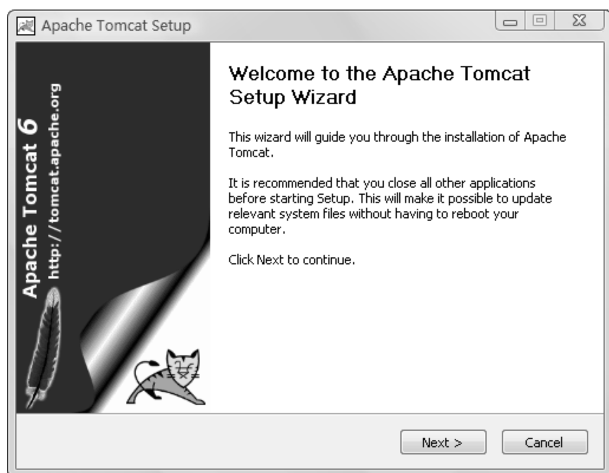


图 7-4　“Apache Tomcat Setup”窗口

2）在打开的新窗口中，单击“I Agree”按钮，如图 7-5 所示。

3）在打开的新窗口中，单击“Next”按钮，将会打开如图 7-6 所示的界面。

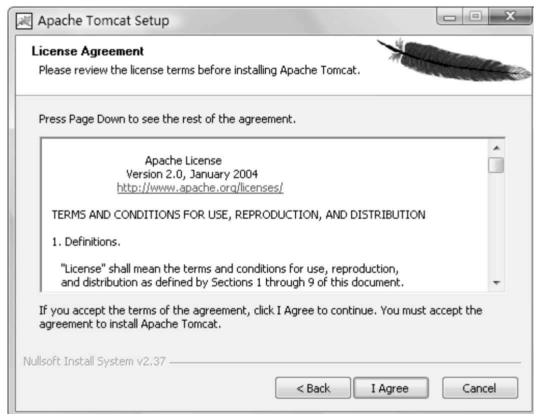


图 7-5 “License Agreement”窗口

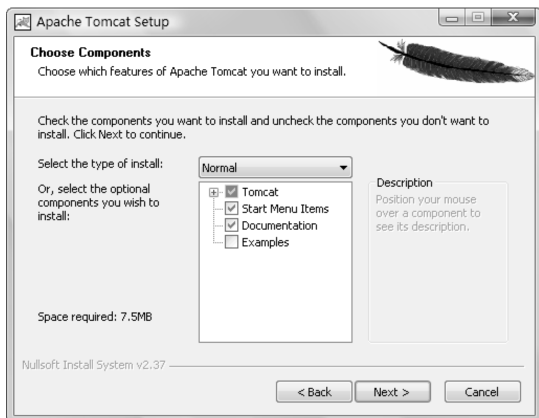


图 7-6 “Choose Components”窗口

4）在打开的新的窗口中要求用户选择安装的位置，用户可以按照默认，也可单击“Browse”按钮对地址进行修改，假设安装在“C:\Program Files\Java\Tomcat 6.0”中，安装位置界面如图 7-7 所示。

5）在新打开的窗口中，用户可以设置端口，Tomcat 的默认端口是 8080，在此保持不变。然后设置管理用户名的密码，如图 7-8 所示。

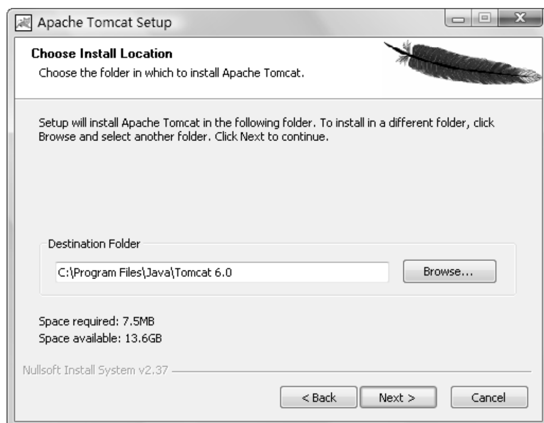


图 7-7 安装目录

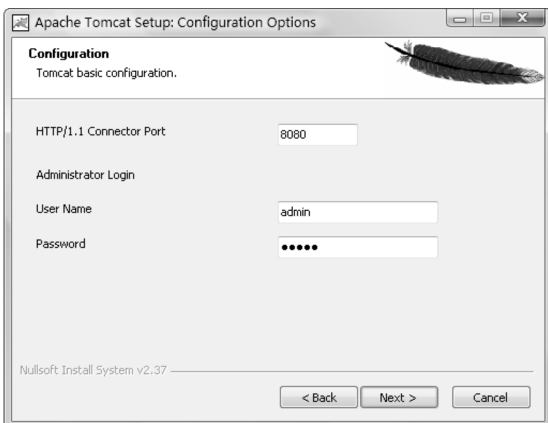


图 7-8 设置 Tomcat 服务器

6）在打开的窗口中选择 JDK 里的 JRE 文件，在默认情况下，它会寻找到这个文件，如果寻找不到，则需要对它进行专门设置，设置好后，单击“Install”按钮进行安装，如图 7-9 所示。

7）安装完成后，将会打开如图 7-10 所示的窗口，用户单击“Finish”按钮即可。

7.1.3 启动 Tomcat

当安装 Tomcat 完成后，就可以启动运行它，具体操作流程如下。

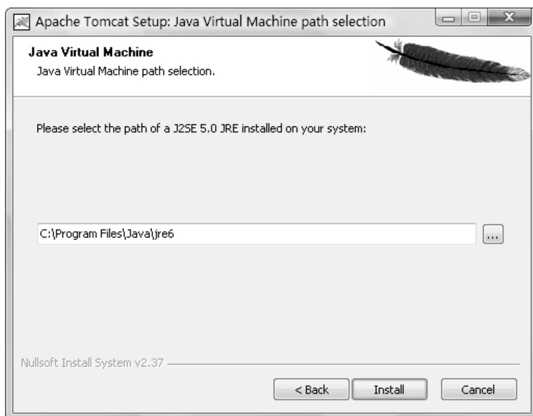


图 7-9　“Java Virtual Machine”窗口

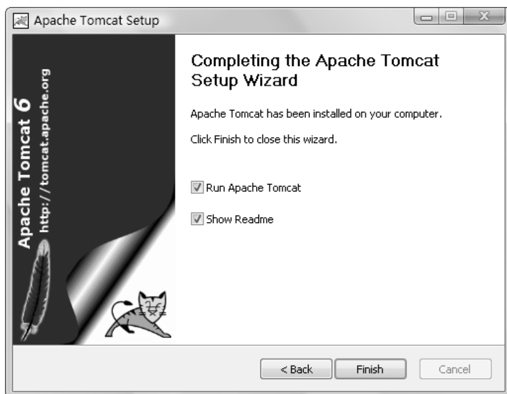


图 7-10　安装完成

1）选择 Tomcat 程序，在任务栏中双击 Tomcat 图标，单击“Start”按钮，如图 7-11 所示。

2）在浏览器中输入测试地址，如图 7-12 所示。

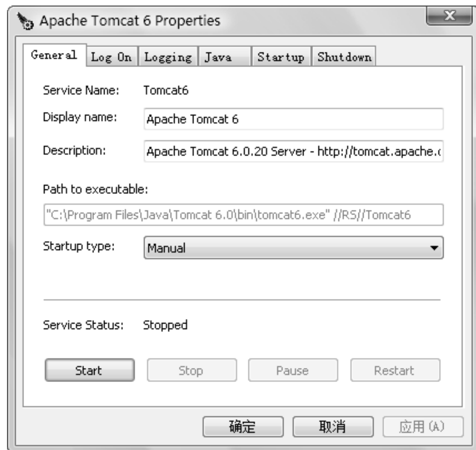


图 7-11　启动 Start



图 7-12　浏览器页面

7.1.4　设置 Tomcat 虚拟目录

设置虚拟目录是开发项目时必不可少的步骤。设置过程十分简单，其具体操作如下。

1）在任意一个盘中新建一个文件夹，例如，在 D 盘新建一个名为“site”的文件夹，复制“Tomcat6.0\webapps\ROOT”目录下的“WEB-INF”文件夹到新建的“D:\site”目录下，如图 7-13 所示。

2）打开“D:\site\WEB-INF”目录下的文件 web.xml，在</description> 之后加入下面的代码。

```
<!--JSPC servlet mappings start -->
<!--JSPC servlet mappings end -->
```

如图 7-14 所示。

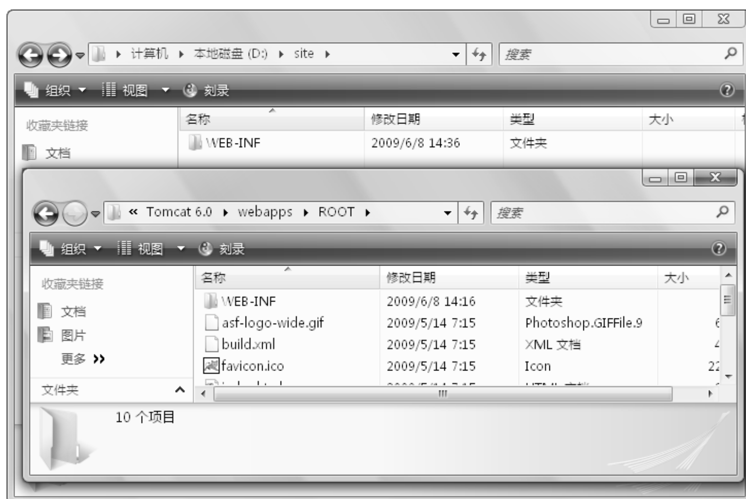


图 7-13 复制 WEB-INF 文件夹

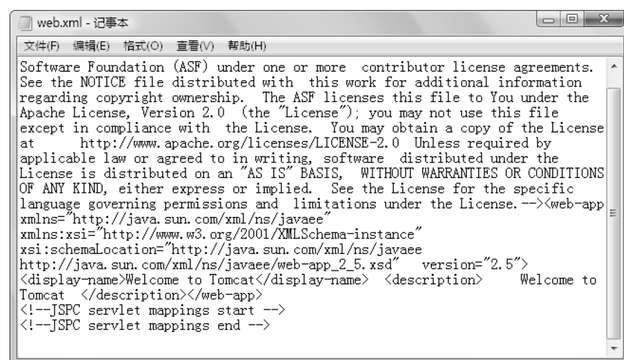


图 7-14 配置 web.xml 文件

3）打开文件 Tomcat6.0\conf\server.xml，在 <Host> 和 </Host> 之间加入下面的代码。

```
<Context path="/site" docBase="d:\site"></Context>
path="/site"
docBase="d:\site">
```

如图 7-15 所示。

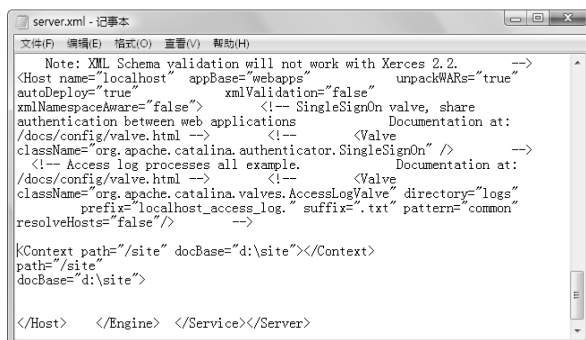


图 7-15 修改 sever.xml

4）打开文件 Tomcat6.0\conf\web.xml，找到下面的代码。

```
<init-param>
<param-name>listings</param-name>
<param-value>false</param-value>
</init-param>
```

把 false 修改为 true 并保存，重启 Tomcat，现在就可以使用目录“http://localhost:8080/site”作为虚拟目录了，如图 7-16 所示。

5）新建一个记事本文档，在记事本文档里输入“虚拟目录配置成功”，然后将其保存，修改它的文件名和扩展名，修改成“index.html”，运行此文件后的效果如图 7-17 所示。

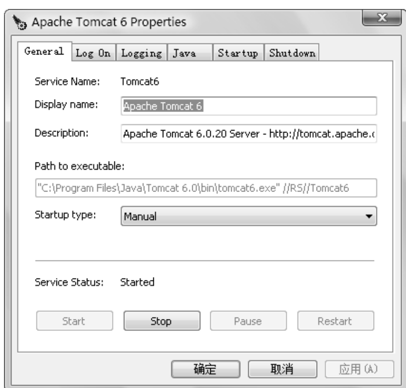


图 7-16　重启 Tomcat

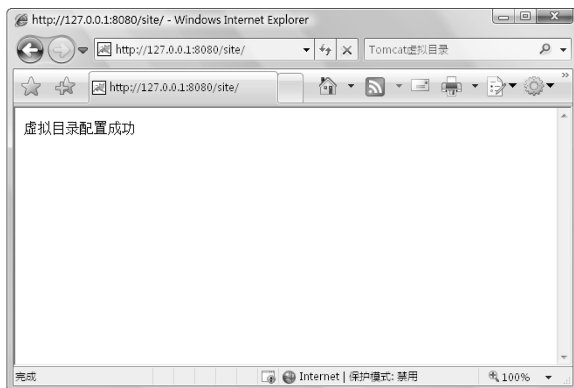


图 7-17　测试页面

7.1.5　一个简单的 Java Web 程序

新建服务器环境与设置好虚拟目录后，就可以编写 Java Web 程序了，具体操作流程如下。

1）新建一个记事本文档，在记事本文档里输入如下程序，其代码如下。

```
<% String Str = "欢迎使用 Java Web";
out.print("大家好！好！ 好的很!! "); %>
<h2><%=Str%></h2>
```

将上述代码保存，然后修改它的文件名和扩展名，修改成“1.jsp”，如图 7-18 所示。

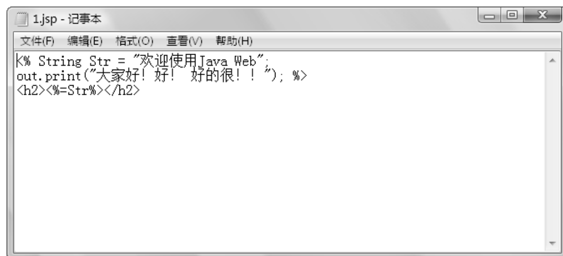


图 7-18　书写的第一个 Java Web 程序

2）在浏览器的地址栏中输入浏览页面的地址，如图 7-19 所示。



图 7-19 第一个 Java Web 页面

7.2 下载和安装 MyEclipse

Eclipse 是开发 Java 的一个利器，MyEclipse 是 Eclipse 的收费插件，主要用于开发企业级的程序。为了给编程者带来方便，在新版本的 MyEclipse 中将 Eclipse 融入其中，所以直接下载 MyEclipse 即可同时使用 MyEclipse 和 Eclipse 的功能。在本节的内容中，将详细讲解安装并使用 MyEclipse 的方法。

7.2.1 下载 MyEclipse

MyEclipse 是收费的开发工具，但是读者可以下载试用版本进行学习。并且有一个版本实现了和 Eclipse 的集成，用户只需要直接下载 MyEplipse 后即可使用。当前最常用的版本是 7.0，所以本书将以 MyEclipse 7.0 为例进行讲解。在如图 7-20 所示的 MyEclipse 官方下载界面中选择复选框，然后进入下载页进行下载即可。



图 7-20 下载 MyEclipse

7.2.2　安装 MyEclipse

下载后将它进行安装，安装的过程十分简单，用户只需要将下载的程序双击，然后按照向导即可对它进行安装，其具体操作如下。

1）双击下载的 MyEclipse 7.1，弹出如图 7−21 所示的对话框，安装这个程序的时候要耐心等待，因为安装过程较慢。

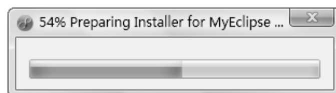


图 7−21　安装 MyEclipse 的进度

2）进度条完成 100%后会弹出另一个进度条对话框，我们不需要管它，等待它完成后会弹出如图 7−22 所示的对话框，单击按钮 Next > 。



图 7−22　开始安装

3）弹出另一个窗口，此时勾选复选框，然后单击按钮 Next > ，如图 7−23 所示。

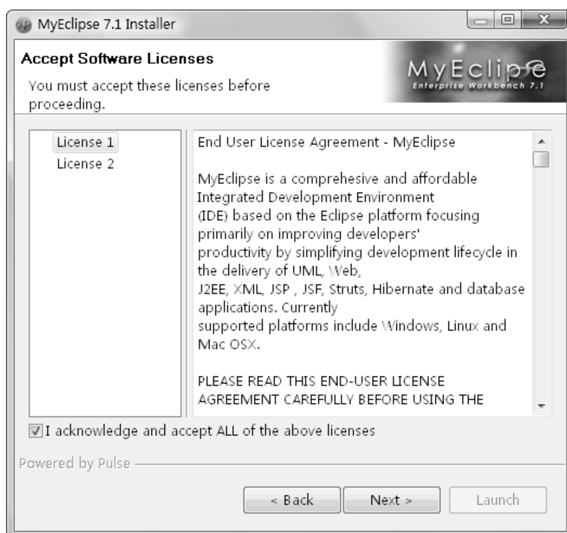


图 7−23　同意协议

4）在新弹出的窗口中单击“Configure”按钮，又弹出一个窗口，在该窗口中设置安装路径，设置完成后，单击“Next”按钮，返回到原来的窗口，单击按钮 Install ，开始进行安装，如图 7-24 所示。

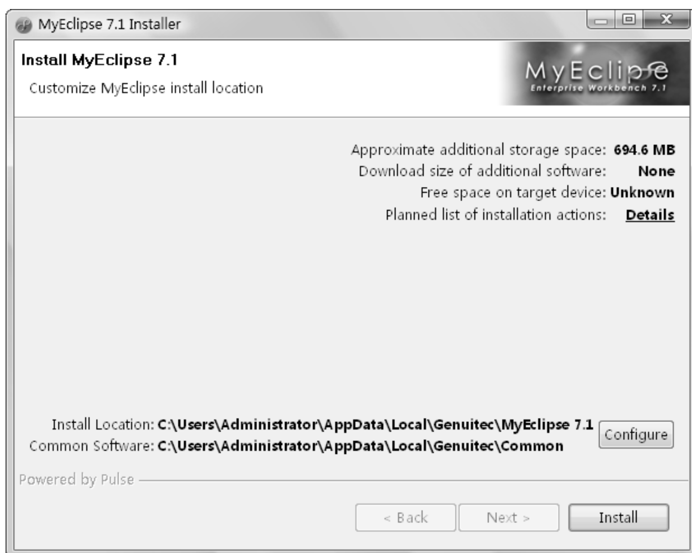


图 7-24　设置安装路径

5）单击过后要等待相当长一段时间才能安装完成，如图 7-25 所示。



图 7-25　等待安装完成

6）安装完成后，要求用户将项目文件设置在一个硬盘目录。笔者在此设置的是 D 盘的 site 文件夹，此文件夹也是 Tomcat 建立的虚拟目录，安装完成后，将会显示如图 7-26 所示的欢迎界面。



图 7-26　欢迎界面

7.2.3　新建一个 MyEclipse 项目

完成 MyEclipse 安装后，将会弹出一个欢迎界面，关闭后可以尝试新建一个项目，其具体操作步骤如下。

1）在 MyEclipse 顶部菜单中依次选择“file/new/web project”命令，如图 7-27 所示。

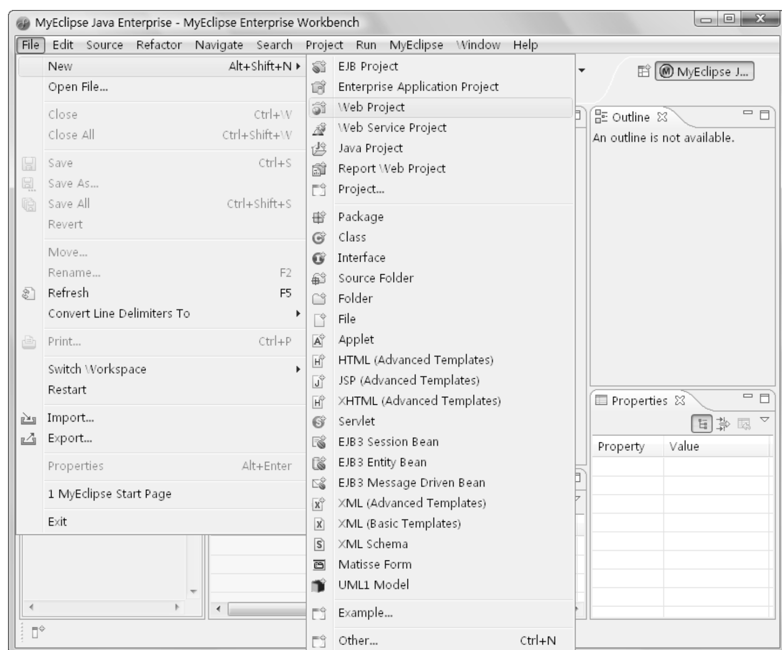


图 7-27　选择命令

2）在弹出“New Web Project”的窗口中可以根据自己的需要进行设置，设置完成后单击“Finish”按钮，如图 7−28 所示。

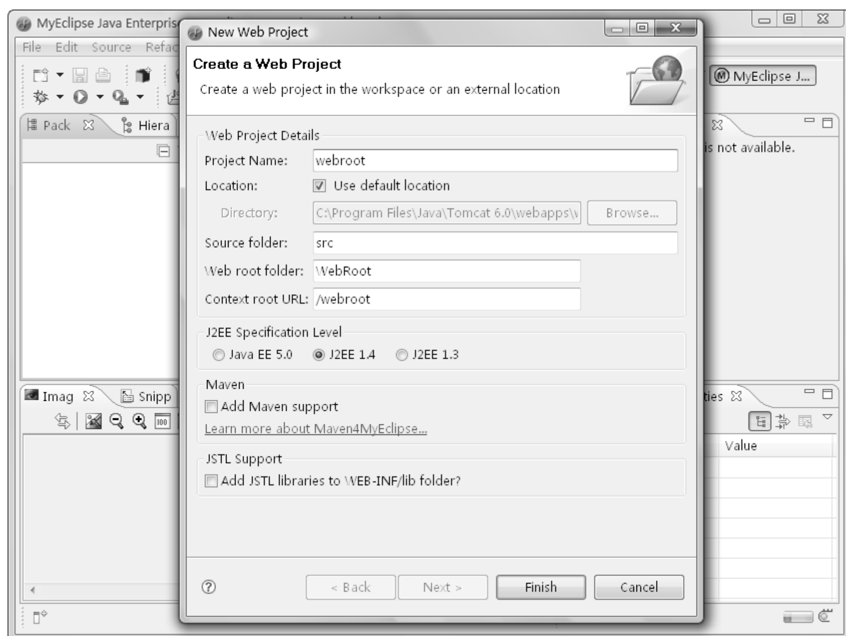


图 7−28　创建项目

3）在启动界面中，选择展开项目，选择“webroot”文件夹，单击鼠标右键，选择“new”命令，用户可以根据需要创建不同的文件，如图 7−29 所示。

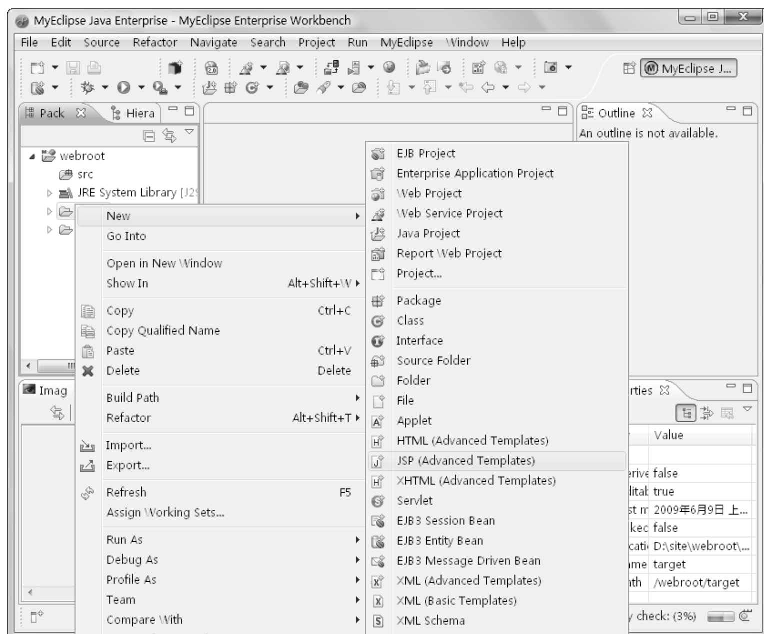


图 7−29　新建文件

7.2.4　部署开发环境

安装了 MyEclipse 和 Tomcat 后，还需要经过部署之后才能成功运行代码。部署开发环境的具体步骤如下。

1）启动 MyEclipse 和 Tomcat 后，选择“MyEclipse/preferences”命令，如图 7-30 所示。

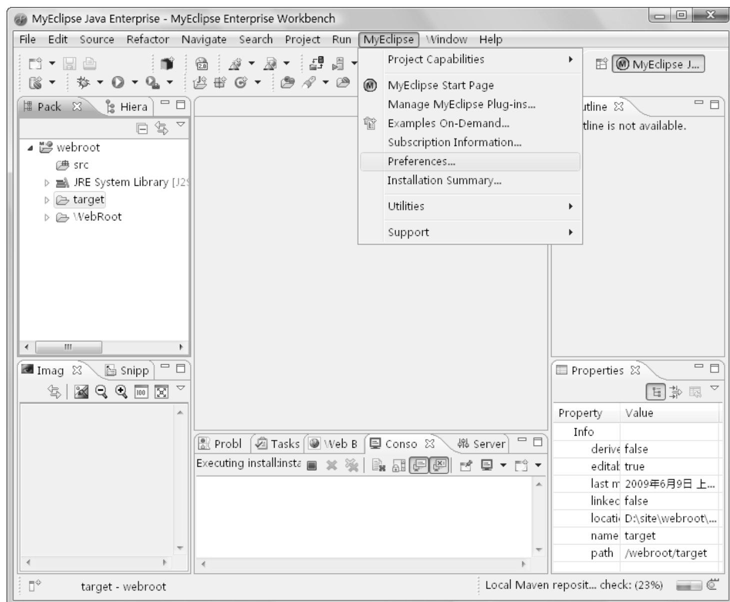


图 7-30　选择命令

2）弹出“Preferences”窗口，在 Server 树状目录下选择“Tomcat”选项，在“Tomcat 6.x”面板设置参数后，得到如图 7-31 所示的界面。

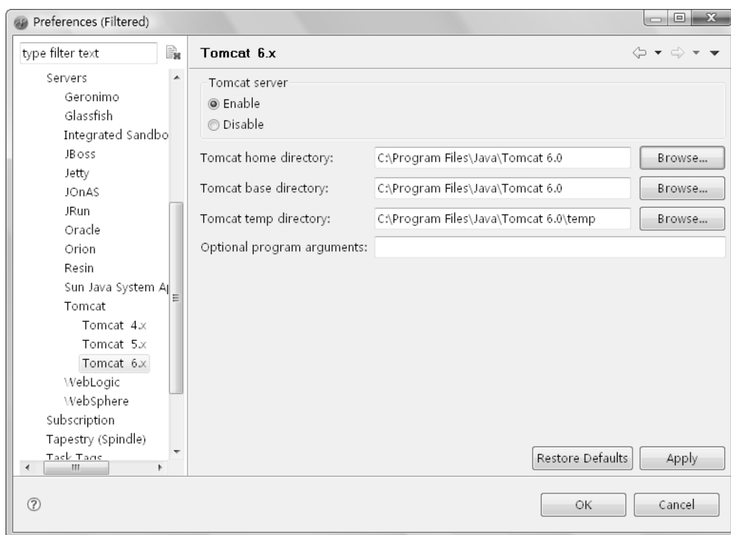


图 7-31　设置 Tomcat 的位置

7.2.5 新建并运行一个 Java Web 页面

当 Java 的运行环境和开发环境搭建完成后，就可以新建一个 Java 文件实现运行调试，具体调试流程如下。

1）选择新建的项目，单击鼠标右键，在弹出快捷菜单中选择“New/JSP”命令，如图 7-32 所示。

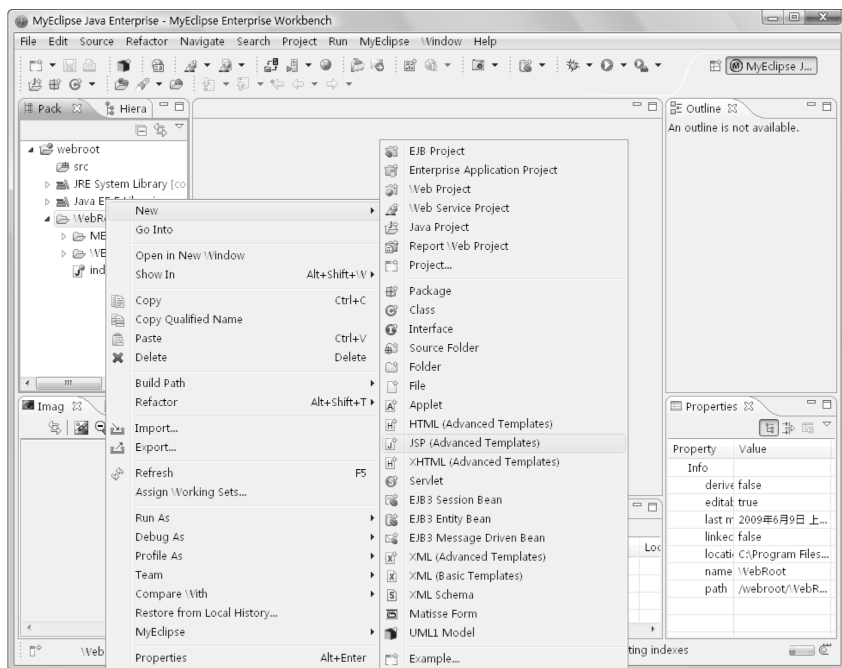


图 7-32 选择命令

2）在弹出的“JSP Wizard”窗口中输入 Java Web 的名称，如图 7-33 所示。

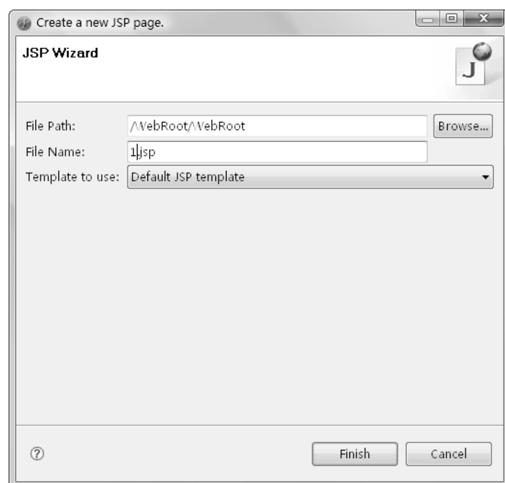


图 7-33 “JSP Wizard”窗口

3）在打开的编辑页面中可以根据视图进行编辑，然后开始编写代码，编写完成后的效果如图 7-34 所示。

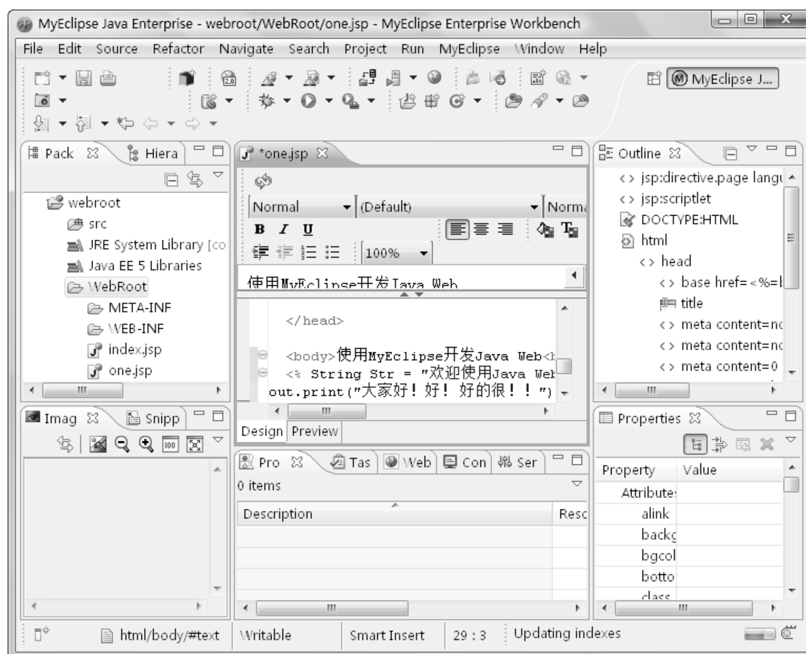


图 7-34　编写代码

4）在打开的“Run As”窗口中选择第三个选项，然后单击“OK”按钮，如图 7-35 所示。

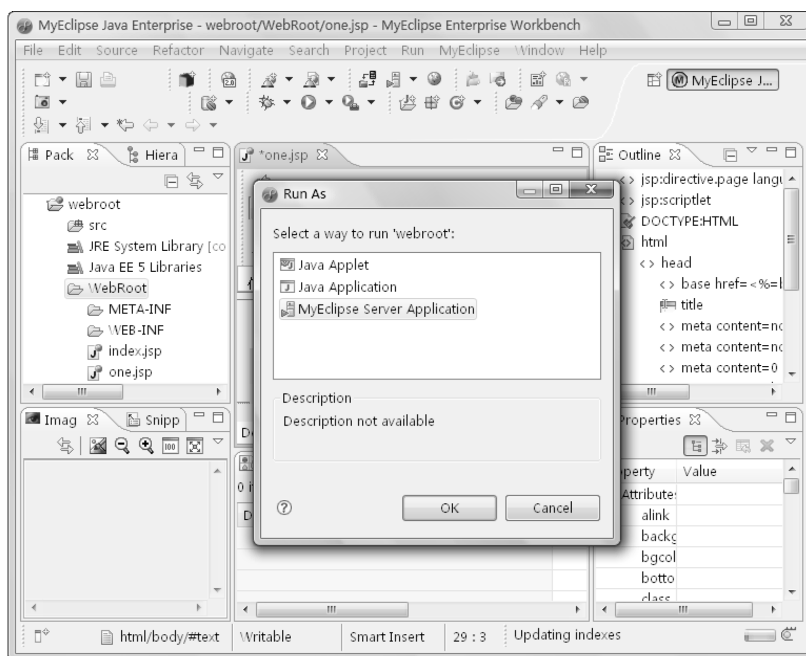


图 7-35　“Run As”窗口

5）打开新的窗口，因为 MyEclipse 部署了服务器，只需要选择第二个服务器，编写的程序就可以在软件中进行调试，单击“OK”按钮，如图 7-36 所示。

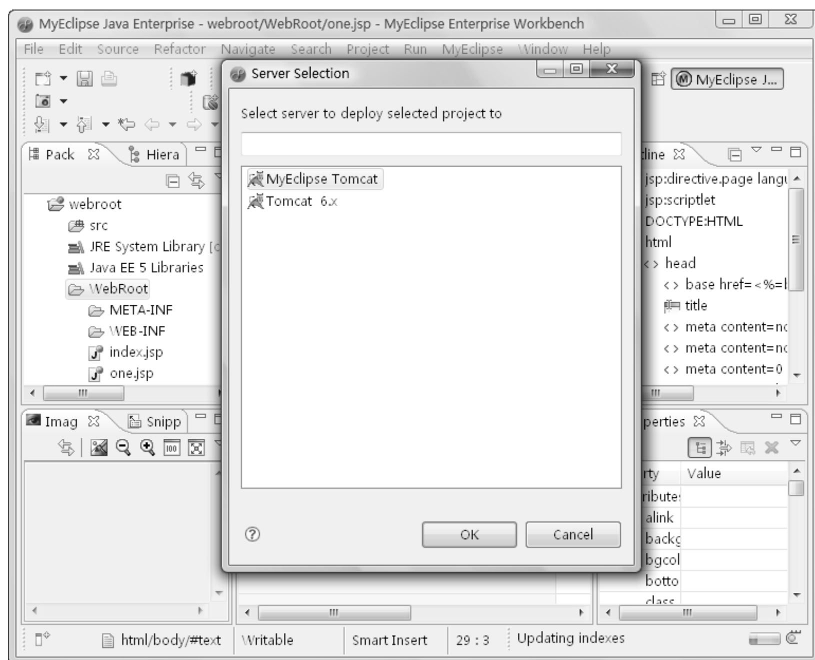


图 7-36 “Server Selection”窗口

6）在浏览器中浏览自己编写的程序，如图 7-37 所示。

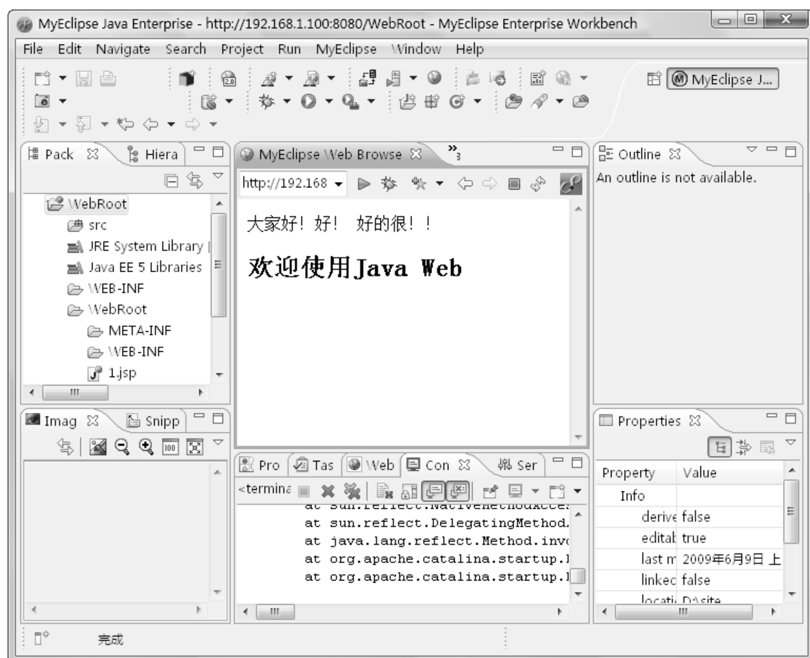


图 7-37 浏览的结果

7.3　疑难问题解析

在本章的内容中，详细讲解了搭建 Java Web 的运行和开发环境的基本知识。本节中，将对本章中比较难以理解的问题进行讲解。

读者疑问：在前面的章节中讲了许多软件，大多是英文的，我英文不好，该怎样熟练使用这些软件命令呢？

解答：这个不要紧，你可以下载一个金山词霸，也可以使用在线的雅虎翻译，通过这些工具能让你很快了解英文的意思。

读者疑问：开发项目时，为什么要设置虚拟目录，它的优点是什么？

解答：在开发项目时，常常需要将所有需要的文件放在一起，并且在开发调试的过程中，为了能够正常调试，必须先建立目录，让它既符合运行环境的条件，又能方便地管理文件。

7.4　职场点拨——一份简历的思考

下面是我的简历。

个　人　简　历

(Personal Particulars)

➢ 个人概况 (Personal　Data)

姓　　名：- 小菜
出生年月：- 1984 年 11 月 01 号
性　　别：- 男
籍　　贯：- ×星
毕业院校：- ××大学
专　　业：- 计算机
学　　历：- 本科
联系方式：- TEL:　152××××××××
E—mail：- ××××××××@yahoo.com.cn

➢ 专业技能（Basic Skills）

擅长计算机维护、应用和开发。
熟悉 ASP、PHP、JSP、C、C++、Visual Basic、C#、JavaScript、HTML、CSS、Java
熟悉 Access、SQL Server、MySQL 数据库，精通 SQL 语言。

其他：熟悉 Dreamweaver、Phtoshop、CAD、Flash，精通 Linux、UNIX。

➢ 工作经历(Social Practice)

➢ 2005 年 6～9 月 ××××××××××× 公司（在校兼职） 网站管理员
➢ 2006 年 10 月～2007 年 6 月 ××××××××××× 公司 程序员
➢ 2007 年 8 月～2009 年 12 月 ××××××××××× 公司 项目开发

➢ 个人特质 (Personal Specialty)

◆ 性格沉稳，待人诚恳。
◆ 独立能力较强，做事大胆细心，敢于迎接新的挑战。
◆ 对工作和朋友始终充满热情，思维活跃，有较好的团队精神。

写完简历后，我感觉很不错，就将简历发送到了 DD 公司 HR 的邮箱。一周后，我还没接到对方的电话。于是电话咨询，对方客气地回答："对不起，小菜先生，经我们分析后您不适合我公司的这个岗位，欢迎下次来 DD 公司应聘！"当时我的心凉了大半截，第一次投简历就出师不利！人们常说"失败是成功之母"，不灰心，继续投。

终于在一个月后，EE 公司打来了电话，让我去参加面试。

看来小菜的简历很成功，起码有公司接纳了，但事实是这样子吗？

现在看来，当时自己的简历是多么的幼稚。如果换个角度去看问题，看 DD 公司 HR 的一句话：

这篇招聘广告在时报、日报、招聘网中发了一个月之久，没有找到一位合适的。我们的目的很简单，就是招聘一个有开发经验的 Java 程序员，能够熟练、独立完成项目。

原来他们的目的这么简单，只要一个有 Java 开发经验的程序员。因为小菜是应届生，先抛开工作经验。看他的简历：上面罗列了很多和计算机有关的技术，真可谓全才。俗话说"业多则不精"，很少有人能精通多种技术，除非是天才。

这份简历给 DD 公司 HR 的印象是：学得很杂，每门技术都懂一点，但没有很精通的一门技术，不适合本公司的职位。

再看这份简历给 EE 公司 HR 带来的印象：应届生，经验有限，但是懂各项基本编程技术，有培养前途。

所以在入行之前一定要认清自己的目的。除了学习 C 语言、Visual Basic 和 SQL 这些最基础的之外，将来想干什么，要根据目的去学习想要的知识，并精通掌握它，才能在职场中立于不败之地。例如，想步入 Web 开发，就学习 Java Web 和 PHP 等相关知识。不要贪多，要学一门，精通一门。

第 8 章　HTML 和 XML

HTML 和 XML 是 Java Web 的基础，开发 Java Web 项目必须用到 HTML 和 XML 技术。究竟什么是 HTML，什么是 XML，初学者又该如何理解这两种技术？在本章将为读者一一解决上述问题。通过本章能学到如下知识。

- HTML。
- 整齐摆放文字。
- 表格。
- 表单。
- CSS 样式和 JavaScript。
- XML。
- 职场点拨——常见的面试错误。

背景介绍

2009 年 XX 月 X 日，天气阴。

明天要去参加 EE 公司的面试了，我很紧张，担心自己不能通过面试。



一问一答

小菜：“明天要参加 EE 公司的面试了，好紧张啊。”

Wisdom：“嗯，要好好把握啊，本章最后所提到的面试常见错误你一定要避免！”

小菜：“嗯！言归正传，本章的 HTML 和 XML 很重要吗？”

Wisdom：“当前最主要的脚本语言是 HTML 和 XML。例如，你应聘时做了很多准备，而 Java Web 开发也需要具备一些知识才能进行。Java Web 开发和脚本语言有关，虽然脚本语言对 Java Web 来说是那么的微不足道，处于编程语言的最底层，但是所有的页面元素几乎都是通过脚本语言来显示出来的。”

小菜：“明白了，XML 和 HTML 是 Java Web 开发的必备知识！”

Wisdom：“对，HTML 和 XML 都是网页离不开的元素，其中 HTML 是用来表现网页的一种形式，而 XML 是用来存储网页数据内容的一种语言。”

8.1　认识 HTML 和 XML

HTML 和 XML 都是开发网页离不开的基本技术，HTML 是用来表现网页的一种形式，

而 XML 是用来存储网页数据内容的一种语言。下面通过一段代码让读者认识 HTML 的作用，其代码（光盘：源代码/第 8 章/8-1.html）如下。

```
<html>
  <head>
    <title> 标示标记</title>
  </head>
  <body>
    <li>重庆
    <li>上海
    <li>北京
    <li>天津
 <ol type=I>
    <li>预习书本
    <li>认真读书
    <li>复习书本
    </ol>
<DL>
    <DT>性别：<DD>男、女
    <DT>职业 :<DD>工程师、教师、程序员
    </DL>
  </body>
</html>
```

上面的代码十分简单，编写完毕后就可以在浏览器中进行浏览，运行效果如图 8-1 所示。

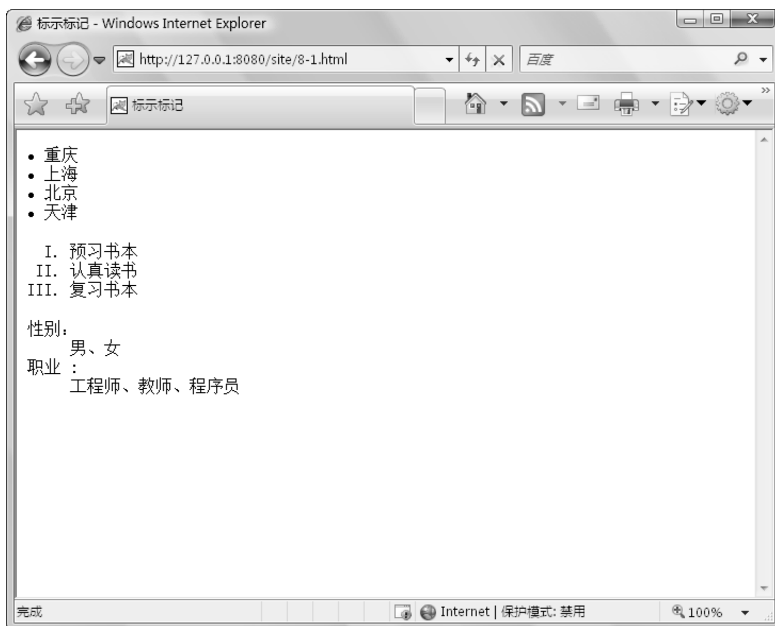


图 8-1　运行后的结果

8.2　理解 HTML

HTML 十分简单，读者只需要理解前后标签和标签的一些参数，就可以记清楚 HTML 的结构。下面通过一段简单的代码来讲解 HTML 结构，其代码（光盘：源代码/第 8 章/8-2.html）如下。

```
<html>
<head>
<title>HTML，相见恨晚</title>
</head>
<body>
<h1>你还好吗，心里一直在呼喊</h1>
<div>很想找到你，很想和你一起，看那云山雾绕的都市……</div>
<body>
</html>
```

网页就是通过这种方式展现给浏览者的，各个参数介绍如下。

- <html>…</html>标签：这是 HTMl 标签，所有标记都要放在这里，<html>是开始标签，</html>是标签的结束。
- <head>…</head>标签：表示网页的头部。
- <title>…</ title >标签：表示网页的标题。
- <h1>…</h1>标签：表示第 1 级标题。
- <div>…</div>标签：层的标签。
- <body>…</body>标签：表示网页的内容。

提示：任何 HTML 标记都是在<html>、<body>和<head>标签中包含的，CSS 样式和 JavaScript 也是包含在这 3 个标签中的。也就是说，CSS 样式和 JavaScript 也是嵌套在 HTML 代码中的。

执行上述代码后会得到如图 8-2 所示的效果。



图 8-2　HTML 的语法

8.3　如何用 HTML 表现字体

在任何一个网页中几乎都离不开字体。在 HTML 中，它是如何管理字体的呢？在本节

将给出这个问题的答案。

8.3.1　标题就是这么酷

在 HTML 中设置标题的方法十分简单，只需使用<h>标签即可实现，它的语法格式如下。

```
<h#> ... </h#>
```

说明：

#代表数字，是从 1 到 6 的整数，数字越大字号就越小。<h#>…</h#> 这些标记显示黑体字，并且在<h#>…</h#> 标记中会自动插入一个空行，不必用<p> 标记再加空行。在同一个<h#>…</h#> 标记中无法使用不同大小的字体。

下面通过一段代码讲解使用标题标记的方法，其代码（光盘：源代码/第 8 章/8-3.html）如下。

```
<html>
<head>
<title>标题也可以这样酷</title>
</head>
<body>
<h1>我是一级标题</h1>
<h2>我是二级标题</h2>
<h3>我是三级标题</h3>
<h4>我是四级标题</h4>
<h5>我是五级标题</h5>
<h6>我是六级标题</h6>
<body>
</html>
```

执行上述代码后会得到如图 8-3 所示的效果。

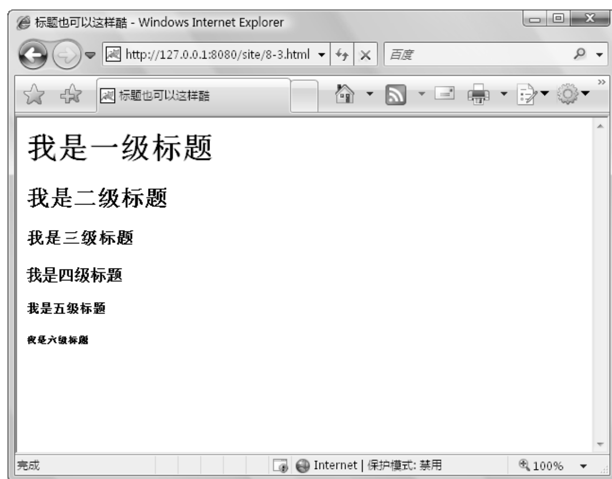


图 8-3　标题的设置

8.3.2　普通文字的大小、颜色和标记

设定文字的大小、颜色和字形标记都是网页开发中的常见任务，几乎所有网页都会设置这 3 种属性。下面通过一段 HTML 代码，讲解设定文字的大小、颜色和字形标记的过程，其代码（光盘：源代码/第 8 章/8-4.html）如下。

```
<html>
  <head>
    <title>瑶琴易抚，相思难寄</title>
  </head>
  <body>
    <font color="#D4200" size="5" face="隶书">斜晖脉脉，海风习习，浪拍礁石，
溅起的水花湿了我的脸，你拂袖替我擦去。</font>
  </body>
</html>
```

执行上述代码，得到如图 8-4 所示的效果。

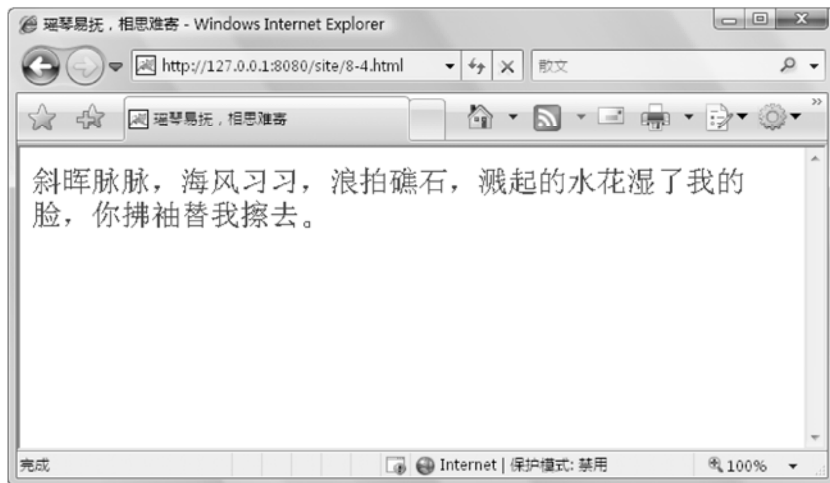


图 8-4　设置普通字体

8.3.3　将文字加粗、倾斜和加底线

在网页中，将字体加粗、倾斜和加底线是常有的情况，在 HTML 中可以通过一些标签来实现这些功能。例如，在下面的 HTML 代码中，分别实现了将文字加粗、倾斜和加底线的效果。其代码（光盘：源代码/第 8 章/8-5.html）如下。

```
<html>
  <head>
    <title>欲将心事付瑶琴，知音少，弦断有谁听</title>
<body>
欲将心事付瑶琴，知音少，弦断有谁听
```

```
        <br>
        </br>
        <b>欲将心事付瑶琴，知音少，弦断有谁听</b>
        <br>
        </br>
        <I>欲将心事付瑶琴，知音少，弦断有谁听</I>
        <br>
        </br>
        <u>欲将心事付瑶琴，知音少，弦断有谁听/u>
        <br>
        </br>
        <body>
        </html>
```

执行上述代码，得到如图 8-5 所示的效果。

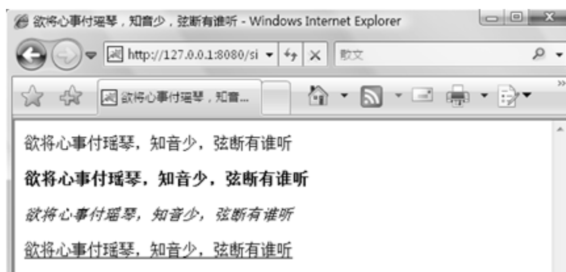


图 8-5　文字标记

提示：这些特殊的标记，用户只能凭自己的记忆将它记住，它没有任何技巧可言，例如，如果在代码中出现了
标签，它的作用是换行，并没有其他特殊用途。

8.4　整齐摆放文字

在网页中，除了对文字进行设置大小、颜色等格式外，还可以让文字根据需要进行排版。在本节的内容中，将一一讲解排版网页元素的基本知识。

8.4.1　换行与不换行

在本书前面的内容中已经讲过换行标签，即
标记，能够实现强制换行功能。如果在 HTML 网页中设置不换行，即使超过浏览器的宽度也不会换行，此时可以使用不换行标签来实现，即<nobr>…</nobr>。下面通过一段代码来讲解换行与不换行功能的实现，其代码（光盘：源代码/第 8 章/8-6.html）如下。

```
        <html>
          <head>
            <title>换行与不换行</title>
        <body>
```

```
        你好吗？<p>很好。
        你好吗？<br>很好。
        <nobr>
        7 月 1 日以后，在我国境内生产销售的计算机出厂前将预装一款名为“绿坝-花季护航”的绿色上网过滤软件，而进口计算机在我国销售前也将预装该软件。</nobr>
        </html>
```

执行上述代码，得到如图 8-6 所示的效果。



图 8-6　换行与不换行

8.4.2　文字的对齐

在 Office 软件里，可以通过左对齐、右对齐、居中对齐和两端对齐来控制文本的摆放，同样在 HTML 里也可以实现这样的排版功能。下面通过一段代码来讲解实现 HTML 文字对齐的流程，其代码（光盘：源代码/第 8 章/8-7.html）如下。

```
<html>
  <head>
    <title>就这样对齐</title>
<body>
<h3 align=center>大海</h3>
<br>
<h3 align=right>大海</h3>
<br>
<h3 align=left>大海</h3>
<br>
</body>
</html>
```

执行上述代码，得到如图 8-7 所示的效果。



图 8-7　对齐设置

8.4.3 项目与列表

在网页中经常需要实现项目或者列表效果，下面将通过一段代码来讲解在 HTML 网页中实现项目与列表功能的基本流程，其代码（光盘：源代码/第 8 章/8-8.html）如下。

```
<html>
  <head>
    <title>项目</title>
  </head>
  <body>
    <li>中国
    <li>俄罗斯
    <li>美国
    <li>英国
    <li>法国
     <li>德国
 <ol type=I>
    <li>刷牙
    <li>洗脸
    <li>睡觉
    </ol>
<DL>
    <DT>性别：<DD>男、女
    <DT>职业 :<DD>工程师、教师、程序员
    </DL>
  </body>
</html>
```

执行上述代码，得到如图 8-8 所示的效果。

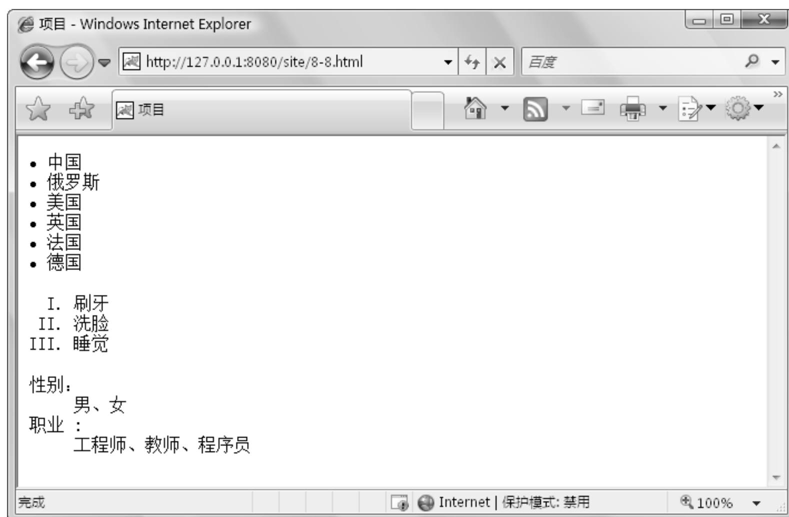


图 8-8 项目与列表

8.5　表格是表现数据的工具

网页离不开表格，它是表现数据的最好工具，是网页实现动态数据交换的基础。在本节的内容中，将对 HTML 表格的基本知识进行讲解。

8.5.1　表格的基本语法

HTML 表格的基本语法很简单，只有 4 个标签，其格式如下。

```
<table>...</table> - 定义表格
<tr> - 定义表行
<th> - 定义表头
<td> - 定义表元(表格的具体数据)
```

实例 25：定义一个表格

本示例的功能是在网页中定义一个简单的表格，其代码（光盘：源代码/第 8 章/8-9.html）如下。

```
<html>
  <head>
    <title>表格</title>
  </head>
  <body>
   <table border>
<tr><th>Food</th>
<th>Drink</th>
<th>Sweet</th>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</table>
  </body>
</html>
```

执行上述代码，得到如图 8-9 所示的效果。

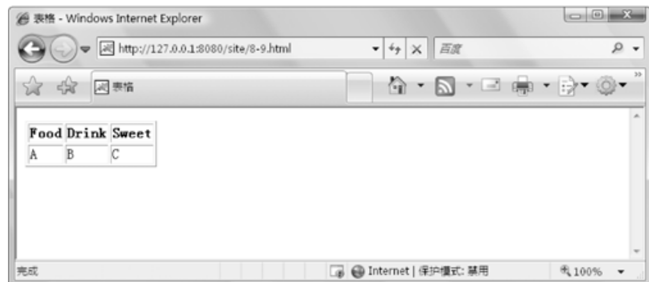


图 8-9　表格



多学一招

上述实例是一个有框的表格，在很多时候，网页需要表现出无框效果的表格样式。实现无框效果的表格样式十分简单，例如，下面的代码（光盘：源代码/第 8 章/8-10.php）。

```
<html>
  <head>
    <title>表格</title>
  </head>
  <body>
   <table>
<tr>
<th>Food</th>
<th>Drink</th>
<th>Sweet</th>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</table>
  </body>
</html>
```

执行上述代码，得到如图 8-10 所示的效果。

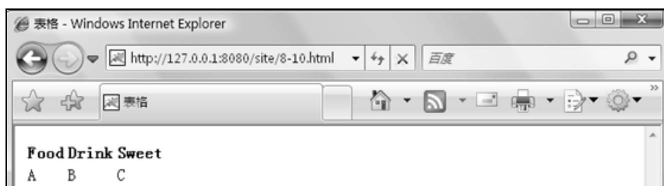


图 8-10　无框表格

8.5.2　跨多行或跨多列的表格

为了表现数据，常常需要实现跨多行或跨多列的效果，下面将通过一个实例来讲解具体实现流程。

实例 26：实现一个跨列表格

在本实例中实现了一个跨多列的表格效果，其代码（光盘：源代码/第 8 章/ 8-11.html）如下。

```
<html>
  <head>
    <title>表格</title>
  </head>
  <body>
```

```
    <table border>
   <tr>
   <th colspan=3> Morning Menu</th>
   <tr><th>Food</th>
   <th>Drink</th>
   <th>Sweet</th>
   <tr>
   <td>A</td>
   <td>B</td>
   <td>C</td>
   </table>
     </body>
   </html>
```

执行上述代码，得到如图 8-11 所示的效果。

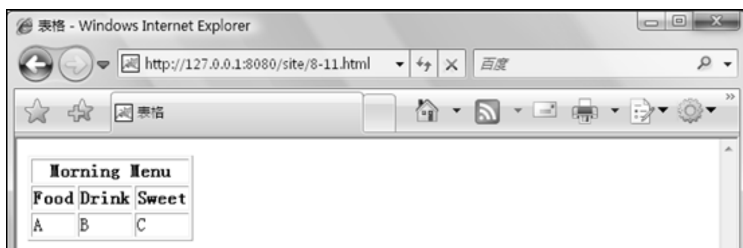


图 8-11　跨多列的表格



多学一招

实例 26 实现了一个跨多列的表格，在很多时候需要在表格中实现跨多行效果，实现方法十分简单，代码（光盘：源代码/第 8 章/8-12.html）如下。

```
   <html>
     <head>
       <title>表格</title>
     </head>
     <body>
      <table border>
   <tr>
   <th rowspan=3> Morning Menu</th>
          <th>Food</th>
   <td>A</td></tr>
   <tr>
   <th>Drink</th>
   <td>B</td>
   </tr>
   <tr><th>Sweet</th>
    <td>C</td>
```

```
        </tr>
        </table>
          </body>
        </html>
```

执行上述代码，得到如图 8-12 所示的效果。

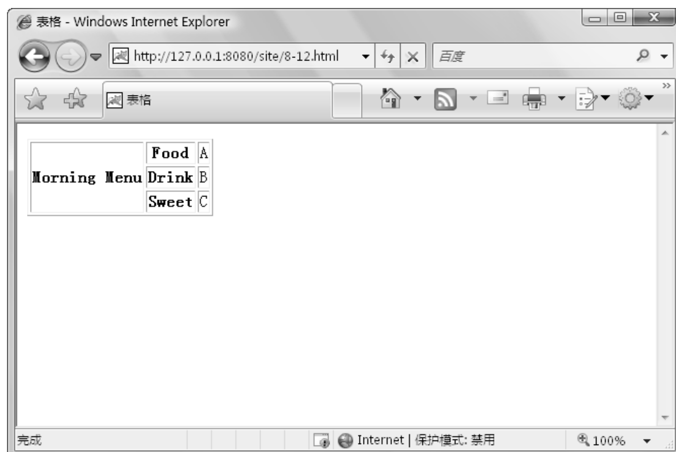


图 8-12　跨多行的表格

8.6　表单让网页实现了交互

在制作动态网页时，表单显得十分重要，它是服务器和浏览器交互的必备手段，本节将详细讲解制作表单的方法。

8.6.1　表单容器<form>

要建立表单必须具备 form 容器，只有在建立 form 容器后才能在里面建立各个组件。在 HTML 中使用<form >…</form>标记表示表单的容器。也许读者对容器的概念十分模糊，我们可以把它想象成一个碟子，里面可以放各种水果（表单）。下面通过一段 HTML 代码来讲解 form 容器的使用流程，其代码如下。

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>表单</title>
</head>
<body>
form 容器
<form id="form 的 ID 号" name="表单名称" method="提交方法" action="处理表单页面">
</form>
</body>
</html>
```

参数介绍。

- <form >：表单容器的标记。
- id="form1" ：表单的 ID 名称，名称是 form1。
- name="form1" ：表单名称。
- method="post"：数据的传送方式。
- action=""：传送页面的设置，用户可以设置一个 Java 的 Web 页面用来处理这个信息。

8.6.2　单行文本框

单行文本框是表单中常见的一种元素，例如，在登录页面中填写一些信息时会用到。下面代码将创建一个单行文本框，其代码（光盘：源代码/第 8 章/8-14.html）如下。

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GB2312" />
<title>表单</title>
</head>
<body>
<form action= method=POST>
您的姓名:
<input type=text name=姓名><br>
您的主页的网址:
<input type=text name=网址 value=http://><br>
密码:
<input type=password name=密码><br>
<input type=submit value="发送"><input type=reset value="重设">
</form>

</body>
</html>
```

执行上述代码，得到如图 8-13 所示的效果。

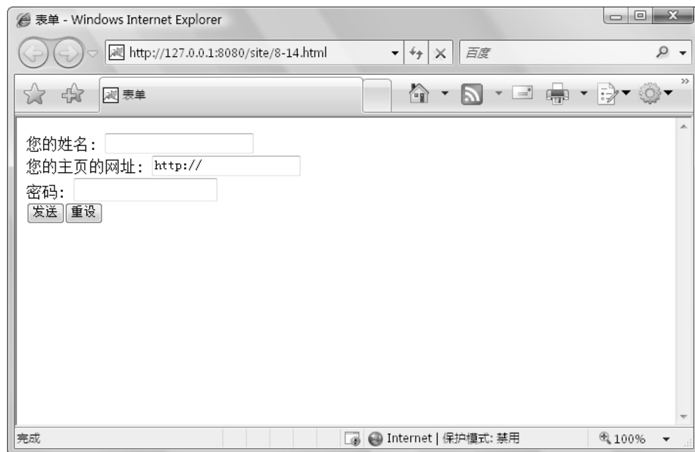


图 8-13　表单

8.6.3 单选按钮

当在文本框中输入信息后，或者在其他表单元素操作后，都需要通过单击按钮来提交信息。我们可以使用单选按钮实现上述功能，究竟单选按钮是如何实现的呢？通过下面的一段代码进行演示，其代码（光盘：源代码/第 8 章/8-15.html）如下。

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GB2312" />
<title>你喜欢吃什么菜</title>
</head>
<body>
<form id="form1" name="form1" method="post" action="">
  <p>
  <input type="radio" name="radio" id="D1" value="D1" />盐煎肉
    <br />
      <input type="radio" name="radio" id="D2" value="D2" />泉水鸡
      <br />
      <input type="radio" name="radio" id="D3" value="D3" />
     回锅肉
  </form>
</body>
</html>
```

执行上述代码，得到如图 8-14 所示的效果。



图 8-14　单选按钮

8.6.4 多行文本和按钮

多行文本框和按钮在表单中的作用举足轻重，它能够接收大段的文字信息，例如，发布日志和留言等信息。下面通过一段代码来讲解多行文本和按钮的使用流程，其代码（光盘：源代码/第 8 章/8-16.html）如下。

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>输入你的文字</title>
</head>
<body>
```

```
<form id="form1" name="form1" method="post" action="">
 <textarea name="Ri" cols="56" rows="10"></textarea>
 <br />
   <input type="submit" name="Tj" id="Tj" value="提交" />
 <input type="reset" name="Tj2" id="Tj2" value="重置" />
</form>
</body>
</html>
```

执行上述代码，得到如图 8-15 所示的效果。

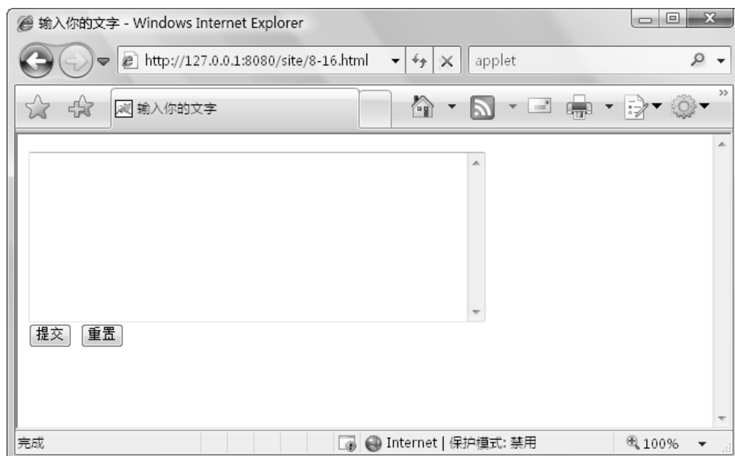


图 8-15　多行文本

8.7　简单认识 CSS 样式和 JavaScript

CSS 样式和 JavaScript 是网页开发中不可缺少的技术元素之一，有了它们的帮助，可以将网页变得更强大、更绚丽。在本节的内容中，将讲解 CSS 和 JavaScript 技术的基本知识。

8.7.1　编写和调用 CSS 样式

先看下面的 HTML 代码。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
   <head>
        <title> 欢迎来到梦之都 </title>
        <link rel="stylesheet" type="text/css" href="dreamdu.css" />
   </head>
   <body>
        <h1>欢迎来到梦之都</h1>
        <p>这是我的第一个网页,在这里
```

```
                <a href="http://www.dreamdu.com/css/">
                    尽情学习 CSS
                </a>吧!
            </p>
        </body>
    </html>
```

将上述文件保存，然后再新建一个名为 dreamdu.css 的文件，在里面输入如下代码。

```
/*段落样式*/
p
{
  color: purple;
  font-size: 12px;
}
/*标题样式*/
h1
{
  color: olive;
  text-decoration: underline;
}
/*链接样式*/
a:link
{
  color:#006486;
}
a:visited
{
  color:#464646;
}
a:hover
{
  color: #ffffff;
  background: #3080CB;
}
a:active
{
  color:white;
  background: #3080CB ;
}
```

执行上面这段代码（光盘：源代码/第 8 章/8-1/），得到如图 8-16 所示的效果。

提示：在制作一个 Web 网站时，如果很多网页都需要这样的 CSS 格式，则很有必要将样式写在一个独立文件里。如果只是在一个文件里需要这个样式，则只需要在〈head〉标签里写下这个样式就可以了，如果对 CSS 代码不懂，可以登录网站“http://www.dreamdu. com/css/”，从零开始学习 CSS 样式的相关知识。因为篇幅的关系，在此不再赘述。

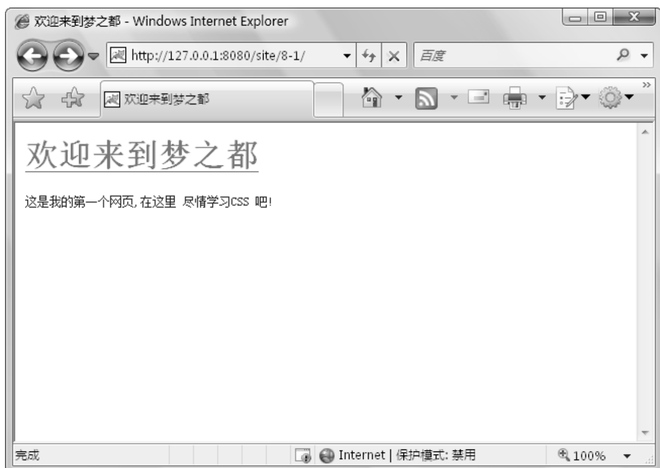


图 8-16　CSS 样式

8.7.2　JavaScript 可以出现在网页的任何位置

JavaScript 可以出现在 HTML 的任意地方。使用标记<script>…</script>后，就可以在 HTML 文档的任意地方插入 JavaScript，甚至在<HTML>之前插入也不成问题。不过如果要在声明框架的网页（框架网页）中插入，就一定要在<frameset>之前插入，否则不会运行。

JavaScript 的基本语法格式如下。

```
<script>
<!—
...
(JavaScript 代码);
...
//-->
</script>
```

每一句 JavaScript 都有类似于以下的格式。

```
<语句>;
```

其中分号“;”是 JavaScript 语言作为一个语句结束的标识符。虽然现在很多浏览器都允许用〈Enter〉键充当结束符号，但是培养用分号作结束符号的习惯是比较科学的。

语句块是用大括号“{ }”括起来的一个或多个语句。在大括号里边是几个语句，但是在大括号外边，语句块是被当做一个语句。语句块是可以嵌套的，也就是说，在一个语句块里可以包含一个或多个语句块。

提示： JavaScript 是一门十分有用的语言，特别在网络特效方面。如果读者要深入学习它，可以登录网站 http://www.dreamdu.com/javascript/下载相关的电子教程。

在动态交互的页面中，通常用 JavaScript 实现验证表单功能，下面通过一个简单的代码来演示上述功能，其代码（光盘：源代码/第 8 章/8-17.html）如下。

```
        <html>
        <head>
        <meta http-equiv="Content-Type" content="text/html; charset=gb2312">
        <title>验证表单元素是否为空</title>
        </head>
        <script language="javascript">
        // 检查表单元素是否为空
        function check(form1){
           for(i=0;i<form1.length;i++){
                if(form1.elements[i].value == ""){       //form1 的属性 elements
的首字母 e 要小写
                     alert(form1.elements[i].name + "不能为空!");
                     form1.elements[i].focus();   //指定表单元素获得焦点
                     return;
                }
           }
           form1.submit();
        }
        </script>
        <body>
        <table width="350" border="1" cellpadding="0" cellspacing="1"
bordercolor="#6F0000" >
          <form name="form1" method="post" action="11.php">
          <tr align="center">
            <td height="24" colspan="2"><span>给我留言</span></td>
          </tr>
          <tr align="center">
          </tr>
          <tr>
            <td width="77" height="22" align="center" >昵称:</td>
            <td width="267"><input name="text" type="text" id="text" size="25"
maxlength="80"></td>
          </tr>
          <tr>
            <td align="center" >留言内容:</td>
            <td><textarea name="textarea" cols="31" rows="5"></textarea></td>
          </tr>
          <tr>
            <td align="center"> </td>
            <td><input type="submit" name="Submit" value="提交" onClick="check
 (form1);">
            <input type="reset" name="Submit2" value="重置"></td>
          </tr>
          </form>
        </table>
        </body>
```

```
</html>
```

执行上述代码，得到如图 8-17 所示的效果。

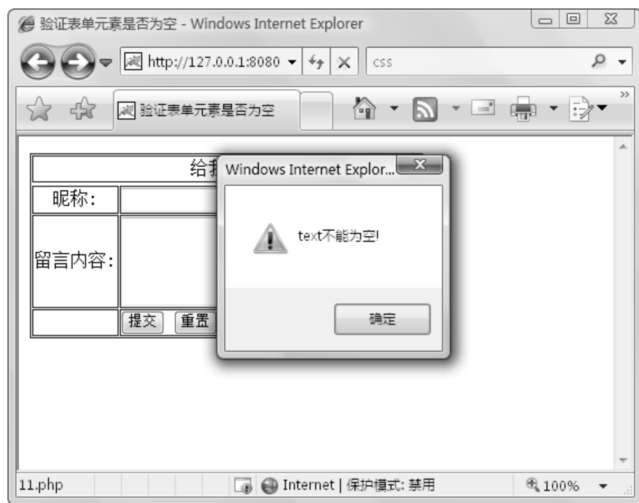


图 8-17　验证表单

8.8　XML

如果读者已经理解了 HTML 的基本知识，那么就不难理解 XML 技术。XML 是一种简单的数据存储语言，使用一系列简单的标记描述数据，而这些标记可以用方便的方式建立。虽然 XML 占用的空间比二进制数据要多，但 XML 极其简单，易于掌握和使用。XML 与 HTML 的区别是：XML 是用来存储数据的，重在数据本身。而 HTML 是用来定义数据的，重在数据的显示模式。

8.8.1　一个简单的 XML

使用 XML 的方法非常简单，它有着固有的格式。下面将通过一个简单的 XML 语句将读者领进门，其代码如下。

```
<?xml version="1.0" encoding="gb2312" standalone="yes"?>
<!--  下面的标签<PHP 编程>就是这个 XML 文档的根目录  -->
<书店>
  <文学类>
        <书名>他的国</书名>
<作者>韩寒</作者>
        <价格>25.00</价格>
<出版社>万卷出版公司</出版社>
        <出版时间>2009-01-01</出版时间>
  </文学类>
</书店>
```

参数介绍。

- <?xml ：表示 XML 声明的开始记号，代表该文件是 XML 格式的文件内容。
- version="1.0" ：XML 的版本说明，因为有它的存在所以该行必须放在首行，这也是声明中不可缺少的内容。尽管以后的 XML 版本可能会更改该数字，但是 1.0 是当前的版本。
- encoding="gb2312" ：是可选项。如果使用编码声明，必须紧接在 XML 声明的版本信息之后，并且必须包含代表现有字符编码的值。默认采用 UTF-8（UTF-8 是 Unicode 的一种变长字符编码，由 Ken Thompson 于 1992 年创建。）编码来解析该文档。
- standalone="yes"：是可选项，独立声明指示文档的内容是否依赖来自外部源的信息。如果使用独立声明，必须在 XML 声明的最后。在文档引用外部 DTD（DTD 是一套关于标记符的语法规则。它是 XML1.0 版规格的一部分，是 XML 文件的验证机制，属于 XML 文件组成的一部分），分析器将报告错误。省略独立声明与包含独立声明 "no" 的结果相同。XML 分析器将接受外部源（如果有）而不报告错误。
- 自定义标签：上述代码中的<书店>等都属于自定义标签，用户可以根据自己需要设置标签，只要成对出现就可以了，十分简单。

8.8.2　XML 的其他元素

XML 其实十分简单，也很容易使用。接下来将讲解 XML 的其他构成元素，以方便读者更好地认识与理解 XML。

1．XML 的注释

XML 是一种标记语言，需要有注释的帮助才能让它的功能更为完善。与其他编程语言类似，注释在文档结构中起着举足轻重的重要，能够给 XML 增加阅读性。注释可以出现在文档序言中(声明之后，根元素之前)，也可以出现在文档之后或文本内容中。但是注释不能出现在属性值中，也不能出现在标记中。XML 在执行时如果遇到“>”，则认为注释已结束，然后继续将文档作为正常的 XML 处理。因此，字符串“>”不能出现在注释中，除了该限制外，任何合法的 XML 字符均可以出现在注释中，这样可以从解析器看到的输出流中删除 XML 注释，同时又不会删除文档的内容。

上面展示了 XML 的注释，注释的各部分含义如下。

- <!--：注释以“<!--” 开头。
- -->：注释以“-->”结尾。

下面通过一段代码讲解 XML 注释的用法，其代码如下。

```
<?xml version="1.0" encoding="gb2312" standalone="yes"?>
<!--  下面是 XML 元素属性的定义！ -->
<aa\>a 是树主干
 <b>b 是树枝
<!--  下面是 XML 元素属性的定义！ -->
<myElement contraction='isn&apos;t'/>
```

```
<!--  下面是 XML 元素属性的定义! -->
</b>
</a>
```

2．XML 的属性

通过属性可以使用名值对的方式（为名称赋予一个值，形似“name=value”）添加与元素有关的信息。属性经常用于为不属于元素内容的元素定义属性，增加元素的描述信息，元素内容由属性值确定。

属性可以出现在开始标记中，也可以出现在空标记中，但是不能出现在结束标记中。其格式如下。

```
<elementName att1Name="att1Value" att2Name="att2Value"...>
```

属性必须有名称和值，不允许没有值的属性名。元素不能包含两个同名的属性。因为 XML 认为属性在元素中出现的顺序并不重要，所以 XML 分析器可能会保留该顺序。

与元素名一样，属性名区分大小写，并且必须以字母或下画线开头。名称的其他部分可以包含字母、数字、连字符、下画线和句点。

下面通过一段代码来演示 XML 属性的基本用法，其实现代码如下。

```
<?xml version="1.0" encoding="gb2312" standalone="yes"?>
<!--  下面是 XML 元素属性的定义! -->
<a>a 是树主干
 <b>b 是树枝
    <myElement contraction='isn&apos;t'/>
    <myElement question="They asked &quot;Why?&quot;"/>
    <myElement contraction="isn't"/>
    <myElement question='They asked "Why?"'/>
    <myElement contraction="isn't" question='They asked "Why?"'/>
 </b>
</a>
```

8.9 疑难问题解析

在本章的内容中，详细讲解了 HTML 和 XML 网页标记语言的基本知识。本节中，将对本章中比较难以理解的问题进行讲解。

读者疑问：HTML 是用来表现网页内容的，那么 CSS 是用来做什么的？在这一章虽然简单提到了 CSS 样式，但不知道它起什么作用？

解答：CSS 样式能够管理网页的显示格式，如网页字体的大小、颜色，在 Web 2.0 后，为了后期的维护，为了代码的简洁，就引进了 CSS 样式，它的合理使用，让网页迈上了新的台阶，从而达到了格式与内容分离，管理更为简单。

读者疑问：在前面几章，学习了 Java、HTML，提到了 Java Web，它们三者是什么关系？

解答：这样讲，Java Web 即 JSP 标记语言，是嵌套在 HTML 中间的，所以用户必须懂 HTML，JSP 的功能十分有限，它根本不能称做 Java Web，但是它有一个很好的工具，就是可以引进 Java 的编译过后的文件.class，这样 Java Web 功能就十分强大。



职场点拨——常见的面试错误

无论是一名正准备步入职场的毕业生，还是一名准备跳槽的程序员，都需要面对面试。既然面试这么重要，所以在面试过程中一定要充分发挥自己的优点，尽量隐藏自己的缺点。在面试中常见的错误如下。

（1）不善于打破沉默

这种错误很常见。也许你不善言谈，也许你心里紧张，只能等待面试官打开话匣。实际上，无论是面试前或面试中，面试者主动致意与交谈，会留给面试官热情和善于与人交谈的良好印象。

（2）向面试官套近乎

不要试图去和面试官套近乎，而是要从面试官的问题中找到面试公司的“需要”，要表现得像这种人才。

（3）为偏见或成见所左右

这种错误也很常见，例如，误认为貌似冷淡的面试官或是严厉或是对应试者不满意，因此十分紧张。如果发现面试官是一位看上去比自己年轻的后辈，就心中怀疑面试官的资历。求职者要做的是不要怀疑，而是要积极面对不同风格的面试官。

（4）慷慨陈词，却举不出例子

在面试中事实胜于雄辩，最忌讳慷慨陈词却举不出例子。应试者要想以其所谓的沟通能力、解决问题的能力、团队合作能力，领导能力等取信于人，唯有举例。

（5）缺乏积极态势

面试官常常会提出或触及一些让应试者难为情的事情。比方说面试官问：您为什么 5 年中换了 3 次工作？为什么离职等。有人可能就会大谈工作如何困难，上级不支持等。

（6）丧失职业风采

很多面试者被问及现所在公司或以前公司时，会诋毁或抨击原来的老板或者公司，甚至大肆谩骂。在众多国际化的大企业中，或是在具备专业素养的面试官面前，这种行为是非常忌讳的。

（7）不善于提问

有些面试者在不该提问时提问，例如，打断面试官谈话而提问。也有些面试者在面试前没有准备好提问问题，轮到有提问机会时不知说什么好。一个好的提问，胜过简历中的无数笔墨，会让面试官刮目相看。

第 9 章　JSP 指令和内置对象

Java Web 最核心的内容是 JSP，JSP 是 Sun 公司倡导的一门动态网页技术标准。初学者可以这样理解 JSP：在 HTML 文件中，加入 Java 程序代码就构成了 JSP 网页。当 Web 服务器访问 JSP 请求的时候，首先要执行其中的 Java 程序代码，然后以 HTML 形式返回。Java 程序运行在服务器端，客户端只用来显示。本章将讲解 JSP 最基本的知识，通过本章能学到如下知识。

- ❑ JSP 语言。
- ❑ JSP 的指令标识。
- ❑ JSP 的内置对象。
- ❑ 职场点拨——谈试用期。

从今天算起，小菜已经步入职场正好满 3 个月，试用期满。

2009 年 XX 月 XX 日，多云

不觉间试用期结束了，我觉得这一段时间表现很好，留用应该没有多大的问题。



一问一答

小菜：“终于试用期满了，下月开始我就是公司的正式员工了！”

Wisdom：“呵呵，这么自信吗？看来表现不错。”

小菜：“当然了，今晚回去好好写一个总结。言归正传，本章的指令和内置对象有什么用？”

Wisdom：“从本章开始，正式步入 JSP 的核心内容，才是真正步入 Web 开发阶段。JSP 是一个简单的标记语言，它和 HTML 十分相似，只是有一些方法，让它变得十分强大。而本章的指令和内置对象又是核心内容中的最基础内容。”

9.1　第一段 JSP 程序

JSP 是一种简单的标记语言，它和 HTML 十分相似。只是有着一些更加强大的功能，能够实现数据的动态显示。下面通过一段代码来演示 JSP 程序，代码（光盘：源代码/第 9 章/9–1.jsp）如下。

```
<%@
   page contentType="text/html;charset=gb2312"
 %>

<html>
     <head>
          <title>网站访问量</title>
     </head>
     <body>
            <h3>欢迎你</h3>
      <%!int i=0;
 %>
<%
  i++;
  out.print(i);
%>个人访问本站
       </body>
</html>
```

上面的代码十分简单，在浏览器中浏览后会得到如图 9-1 所示的效果。



图 9-1　统计流量

9.2　JSP 概述

JSP 是 Sun 公司倡导的一个网页标准，它有着自己独特的特点。在本节的内容中，将详细讲解 JSP 技术的基本知识。

9.2.1　JSP 简介

JSP（Java Server Pages）最初是由 Sun 公司倡导的，后来许多公司纷纷参与，大家一起建立的一种动态网页技术标准。JSP 技术与ASP技术有点类似，它是在传统的网页HTML文件（*.htm,*.html）中插入Java程序段（Scriptlet）和 JSP 标记（tag）形成 JSP 文件（*.jsp）。

用 JSP 开发的 Web 应用是跨平台的，既能在 Windows 下运行，也能在 Linux 等其他操作系统上运行。JSP 使用 Java 编程语言编写类 XML 的 tags 和 Scriptlets，来封装产生动态网页的处理逻辑。网页可以通过 tags 和 Scriptlets 访问存在于服务端的资源的应用逻辑。JSP 将网页逻辑与网页设计和显示分离，支持可重用的基于组件的设计，使基于 Web 的应用程序的开发变得迅速和容易。

当 Web 服务器遇到访问 JSP 网页的请求时，首先执行其中的程序段，然后将执行结果连同 JSP 文件中的 HTML 代码一起返回给客户。插入的 Java 程序段可以操作数据库、重新

定向网页等，以实现建立动态网页所需要的功能。

JSP 与 Java Servlet一样，是在服务器端执行的。因为通常返回该客户端的只是一个 HTML 文本，所以只要客户端有浏览器就能浏览。

JSP 的 1.0 规范的最后版本是 1999 年 9 月推出的，同年 12 月又推出了 1.1 规范。现在最新的规范是 JSP 2.1。

JSP 页面由 HTML 代码和嵌入其中的 Java 代码所组成。当页面被客户端请求以后，服务器会对这些 Java 代码进行处理，然后将生成的 HTML 页面返回给客户端的浏览器。Java Servlet 是 JSP 的技术基础，而且在开发大型 Web 应用程序时，需要 Java Servlet 和 JSP 配合完成。JSP 具备了 Java 技术的简单易用、完全的面向对象、平台无关性、安全可靠和面向因特网的特点。

自 JSP 推出后，如 IBM、Oracle、BEA 等众多大公司都推出支持 JSP 技术的服务器，所以 JSP 迅速成为商业应用的服务器端语言。

9.2.2　JSP 的强势和弱势

JSP 不是万能的，它有自己擅长的，也有自己不擅长的。JSP 的优点如下。

- 一次编写，跨平台到处运行。在这一点上 Java 比PHP更出色，除了系统之外，代码不用做任何更改。
- 系统的多平台支持。基本上可以在所有平台上的任意环境中开发，在任意环境中进行系统部署，在任意环境中扩展。这和 ASP/PHP 相比，其优势是显而易见的。
- 强大的可伸缩性。从只有一个小的Jar文件就可以运行 Servlet/JSP，到由多台服务器进行集群和负载均衡，到多个 Application 进行事务处理和消息处理，一台服务器到无数台服务器，JSP 显示了巨大的生命力。
- 多样化和功能强大的开发工具支持。在进行 Java 开发时，可以使用许多非常优秀的开发工具，而且许多可以免费得到，并且这些工具能够顺利地运行在多种平台上。

其实大家除了掌握它的优点外，还需要了解它的缺点，具体说明如下。

- 由于为了跨平台的功能，为了极度的伸缩能力，所以对应增加了产品的复杂性。
- JSP 的运行速度是用 class 常驻内存来完成的，所以它在一些情况下所使用的内存比起用户数量来说确实是“最低价格比”了。从另一方面，它还需要硬盘空间来储存一系列的.java 文件和.class 文件，以及对应的版本文件。

9.3　认识 JSP 语言

在本章前两节讲解了 JSP 语言的基本知识，JSP 究竟是怎么一回事呢？在本节的内容中，将对 JSP 的基本知识进行更加详细的讲解。

9.3.1　编写第一个 JSP 程序

下面编写一个简单的 JSP 程序，它的内容跟普通 HTML 文件是一样的，唯一的区别是它的文件名后缀是.jsp，而不是 html。这个程序的代码如下。

```
<html>
   <head>
         <title>Welcome</title>
   </head>
   <body>
         <center>First JSP Test</center>
   </body>
</html>
```

如果把上述内容保存为 9-2.html，则可以双击打开这个文件（默认用 Internet Explorer 打开），效果如图 9-2 所示。

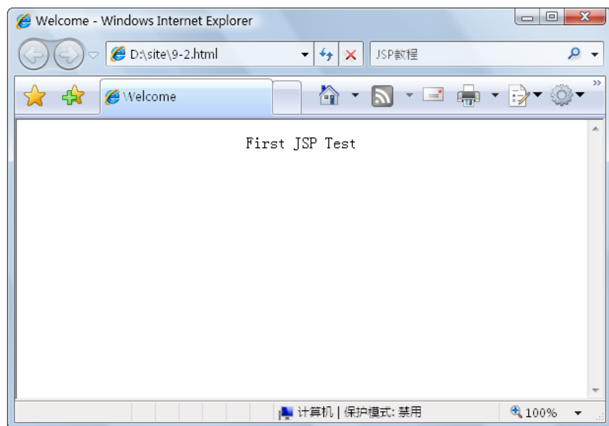


图 9-2 双击打开 HTML 文件

如果把上述内容保存为 9-2.jsp，就不能直接用双击的方式打开这个文件来查看其执行效果。正确的做法是把它发布到 Tomcat 的某个 Web 应用中，只有这样才可以成功运行。例如，把 9-2.jsp 文件复制到对应的目录下，并在浏览器地址栏中输入地址后才可以看到页面的执行效果，如图 9-3 所示。

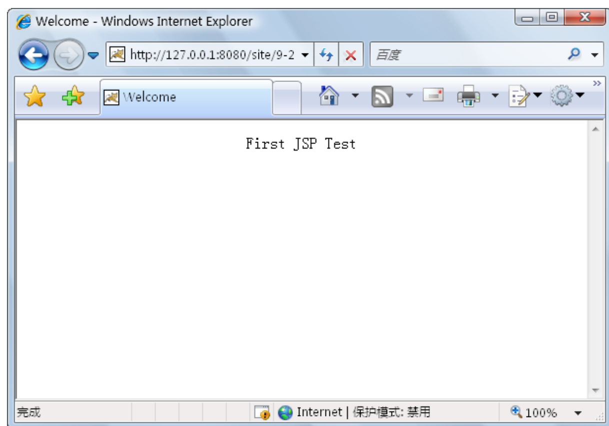


图 9-3 JSP 页面

由执行效果可以看出，两个文件的显示效果是一样的。对于编程而言，它只是把文件的扩展名改为“.jsp”，而这个修改却造成了根本的区别。当以“html”为扩展名时，可以双击使用 IE 浏览器打开它；而以“jsp”为扩展名时，就必须通过 Tomcat（或者其他的 JSP 容器）解析后，才可以看到实际想要看到的效果页面。从这个例子可以看出，HTML 语言中的元素完全可以被 JSP 引擎解析。



事实上，JSP 只是在原有的 HTML 文件中加入了一些具有 Java 特点的代码，这些代码具有其独有的特点，成为 JSP 的语法元素。

9.3.2　了解 JSP 基本语法

JSP 除了 HTML 标签外，还提供了如下 4 种构建网页内容时所需要的元素。

- 指令（Directive）。
- 声明（Declaration）。
- 程序代码（Scriptlet）。
- 表达式（Expression）。

JSP 指令用于设置和整个 JSP 页面相关的属性，如页面的脚本语言，需要导入 Java 包名以及页面编码的字符集等。JSP 指令的语法格式如下。

```
<%@ 指令名 属性=“值”%>
```

JSP 指令包括 page、include 和 taglib3 个指令。

在 JSP 页面中声明方法和属性的方法十分简单，声明的属性和方法适合于 JSP 的任何位置。其中声明属性的格式如下。

```
<%!
int a=2009;
%>
```

声明方法的格式如下。

```
<%!
void method
{};
%>
```

在开发 Java Web 程序时，用标记<% …%>将 JSP 代码与 HTML 代码隔开。换言之，JSP 语言的语法格式如下。

```
<%Java 代码 %>
```

9.3.3　JSP 注释

任何编程语言都有注释，JSP 也不例外。在 JSP 中有多种注释，严格来说，Java 注释也属于 JSP 页面的注释。JSP 的注释分为显示注释和隐藏注释两种，接下来将讲解这两种 JSP

注释的基本知识。

1．显示注释

如果想在客户端显示一个注释，这就需要使用显示注释，下面通过几段代码让读者认识显示注释。

JSP 注释的语法格式如下。

```
<!-- comment [ <%= expression %> ] -->
```

例子 1：

```
<!-- This file displays the user login screen -->
```

将在客户端的 HTML 源代码中产生和上面一样的数据。

```
<!-- This file displays the user login screen -->
```

例子 2：

```
<!-- This page was loaded on <%= (new java.util.Date()).toLocaleString()
%> -->
```

将在客户端的 HTML 源代码中显示为。

```
<!-- This page was loaded on January 1, 2000 -->
```

这种注释方法和 HTML 中的很像，可以在“查看源代码”中看到。唯一有些不同的就是，可以在这个注释中用表达式（例子 2 所示），这个表达式是不定的，由页面的不同而不同。我们能够使用各种表达式，只要是合法的就行。

2．隐藏注释

可以将隐藏注释写在 JSP 程序中，但不是发给客户。其语法格式如下。

```
<%-- comment --%>
```

下面通过一段代码进行讲解，其代码如下。

```
<%@ page language="java" %>
<html>
<head><title>A Comment Test</title></head>
<body>
<h2>A Test of Comments</h2>
<%-- This comment will not be visible in the page source --%>
</body>
</html>
```

用隐藏注释标记的字符会在 JSP 编译时被忽略掉。这个注释在你希望隐藏或注释你的 JSP 程序时是很有用的。JSP 编译器不会对<%--and--%>之间的语句进行编译，它不会显示在客户的浏览器中。

9.3.4　JSP 的表达式

什么是 JSP 表达式？这是认识 JSP 的重要问题。在学习 Java 的时候，曾经学习过 Java 的表达式，其实两者是大同小异的，这里只简单地讲解。

JSP 表达式的语法格式如下。

```
<%= expression %>
```

例子：

```
<font color="blue"><%= map.size() %></font>
<b><%= numguess.getHint() %></b>
```

表达式元素表示的是一个在脚本语言中被定义的表达式，在运行后会被自动转化为字符串，然后插入到这个表达式在 JSP 文件的位置显示。因为这个表达式的值已经被转化为字符串，所以可以在一行文本中插入这个表达式，具体形式和 ASP 的完全一样。

在 JSP 中使用表达式时要记住以下两点。

1）不能用一个分号（“；”）来作为表达式的结束符。

2）同样的表达式用在 Scriptlet 中就需要以分号来结尾。

9.3.5　JSP 的声明语句

和其他语句一样，在 JSP 语句中也需要声明变量和方法。在 JSP 中声明变量和方法十分简单，其格式如下。

```
<%! declaration; [ declaration; ]+ ... %>
```

代码如下。

```
<%! int i = 0; %>
<%! int a, b, c; %>
<%! Circle a = new Circle(2.0); %>
```



一定要声明将要在 JSP 程序中用到的变量和方法，否则程序会出错。可以一次性声明多个变量和方法，只要以“;”结尾即可，当然这些声明在 Java 中必须是合法的。当声明方法或变量时，需要遵循下面的规则。

（1）声明必须以“;”结尾（Scriptlet 有同样的规则，但是表达式就不同了）。

（2）可以直接使用在<% @ page %>中被包含进来的已经声明的变量和方法，不需要对它们重新进行声明，一个声明仅在一个页面中有效。

（3）如果想每个页面都用到一些声明，最好把它们写成一个单独的文件，然后用<%@ include %>或<jsp:include >元素包含进来。

下面通过一段代码来讲解 JSP 声明语句的用法，其代码（光盘：源代码/第 9 章/9−3.jsp）如下。

```
<%@ page import="java.util.*" %>
<HTML>
<BODY>
<%!
Date theDate = new Date();
Date getDate()
{
System.out.println( "In getDate() method" );
return theDate;
}
%>
Hello! The time is now <%= getDate() %>
</BODY>
</HTML>
```

上面的代码十分简单，编译运行后就可以在浏览器中浏览，运行效果如图 9-3 所示。

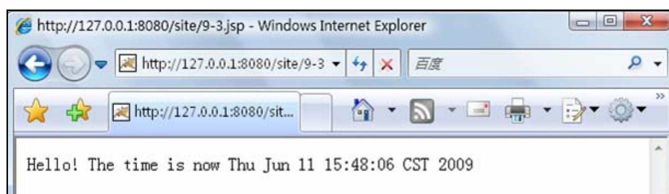


图 9-3　声明

在上述实例中已经声明了变量和方法：一个 Date 变量 theDate 以及一个方法 getDate。声明之后，以上变量和方法就可以在 Scriptlets 和表达式中都有效了，因为它们已经定义。但是上面的这个例子并不能正常工作，不管怎么重载网页，日期都是相同的，原因是这些声明只在网页被装载的时候才被计算一次。读者可以修改上面的例子，增加另外一个函数 computeDate 来重新初始化 theDate。并且增加一个 Scriptlet 来每次调用 computeDate，这样就可以解决上述问题。

9.4　JSP 的指令标识

JSP 指令包括 page、include 和 taglib，用于设置整个 JSP 的属性。本节将详细讲解 Page、include 指令。

9.4.1　page 指令

在 page 指令中，用户可以定义页面的多种属性，例如，脚本语言、编码方式、导入的 Java 包等。page 指令的语法如下。

```
<%@ page
[lang=”java”]
[exteds=”package.class”]
```

```
[import="{package.class|pakcage.*},..."]
[contentType="TYPE;charset=CHARSET"]
[session="True|False"]
[buffer="none|8kb|sizekb"]
[autoFlush="True|False"]
[isThreadSafe="True|False"]
[info="text"]
[isErrorPage="True|False"]
%>
```

各个参数的具体说明如下。

- language="java"：声明脚本语言的种类，默认情况下为 java，可能的其他值有 java、javascript 和 web1。
- import="{package.class|pakcage.*},..."：需要导入包的 java 包的列表“java.lang.*? javax.servlet.*? javax.servlet.jsp.*? javax.servlet.http.*”。因为在编译时已经导入了，所以不需要再次指明。
- contentType="TYPE;charset=CHARSET"：设置 MIME 类型。默认的 MIME 类型是 text/html，默认字符集为 ISO−8859−1。
- session="True|False"：设置客户是否需要 HTTP Session。
- buffer="none|8kb|sizekb"：buffer 的大小以千字节为单位，是 out 对象处理 JSP 页面到客户端浏览器输出时使用的缓冲区。定义的值不小于 8KB，buffer 的默认值为 8KB。
- autoFlush="True|False"：用来设置当 buffer 溢出时，是否需要强制输出。如果其值为 True，则表示输出正常
- isThreadSafe="True|False"：用来设置 JSP 文件是否可以使用多线程。如果设置为 True，那么此 JSP 文件能够同时处理多个用户的请求。默认为 True。
- info="text"：当 JSP 被执行时，用来描述当前 JSP 文件的相关信息。可以通过使用 Servlet.getServletInfo()方法取得。
- isErrorPage="True|False"：设置是否使用 exception 对象

下面将通过一个具体实例来演示使用 page 指令的流程。

实例 27：使用 Page 指令

本实例的目的是讲解 Page 指令的用法，其代码（光盘：源代码/第 9 章/9−4.html）如下。

```
<%@ page import="java.util.*"
   session="true"
   contentType="text/html; charset=GBK"
   buffer="32kb"
 %>
 <html>
     <head>
         <title>我就是 JSP 的 page 指令</title>
```

```
        </head>
        <body>
            <h4>伟大 page 指令，它的作用就是导入一些包使用的</h4>
        </body>
    </html>
```

编写完成上述代码后，就可以在浏览器中对它进行正常的浏览，运行效果如图 9-4 所示。



图 9-4　page 指令



多学一招

除了上面方法可以导入包，还可以导入其他的内容，比如页面的编码格式，这是在 JSP 页面常用的，代码如下。

```
<%@ page language="java" contentType="text/html; charset=GBK"%>
<%@ page language="java" pageEncoding="GBK"%>
```

9.4.2　include 指令

include 指令的作用是将一个页面插入到当前页面，比如有一个 B 页面，用户可以在 A 页面中使用 include 指令，将 B 页面插入在 A 页面。这些文件可以是文本文件、HTML 文件或者 JSP 文件，include 指令的语法如下。

```
<%@ include file="相对路径" %>
```

下面通过两段代码（光盘：源代码/第 9 章/9-1/）来讲解使用 include 指令的流程。

文件 index.jsp 的代码如下。

```
<html>
<head><title>An Include Test</title></head>
<body bgcolor="white">
<font color="blue">
时间是
<%@ include file="date.jsp" %>
</font>
</body>
</html>
```

文件 date.jsp 的代码如下所示：

```
<html>
```

```
        <head>
          <title>test</title>
        </head>
        <body>
            <h4>这个 date 页面的内容</h4>
    <%@ page import="java.util.*" %>
    <%= (new java.util.Date() ).toLocaleString() %>
        </body>
    </html>
```

编码完毕后，可以在浏览器中运行显示执行效果，如图 9-5 所示。

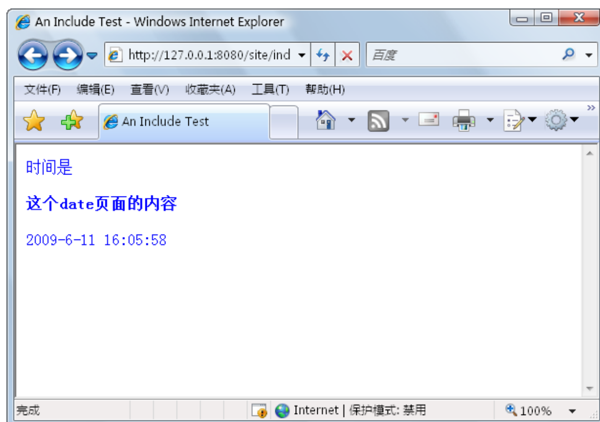


图 9-5　include 指令

提示：

Taglib 指令使用方法与 page、include 指令基本相同，它主要定义一个标签库以及其自定义标签的前缀。定义方法为：<% @ taglib %>指令声明此 JSP 文件使用了自定义的标签，同时引用标签库，也指定了它们的标签的前缀。

9.5　JSP 的内置对象

JSP 提供了一系列的内置对象，这些内置对象不需要先声明，而是由 JSP 容器自动定义，可以在 JSP 的环境中直接使用。在本节的内容中，将详细讲解 JSP 中常用内置对象的基本知识。

9.5.1　Request 对象

在 JSP 中，Requset 对象的作用是给客户发出一个请求，它封装用户所提交的信息，然后通过调用对象相应的方法获取封装的信息。Requset 对象的常用方法如下。

- object getAttribute（String name）：返回指定属性的属性值。
- Enumeration getAttributeNames()：返回所有可用属性名的枚举。
- String getCharacterEncoding()：返回字符编码方式。

- int getContentLength()：返回请求体的长度（字节）。
- String getContentType()：得到请求体的 MIME 类型。
- ServletInputStream getInputStream()：得到请求体中的二进制流。
- String getParameter（String name）：返回 name 指定参数的参数值。
- Enumeration getParameterNames()：返回可用参数名的枚举。
- String[] getParameterValues（String name）：返回包含参数 name 的所有值的数组。
- String getProtocol()：返回请求的协议类型及版本号。
- String getScheme()：返回请求的计划名。
- String getServerName()：返回接受请求的服务器主机名。
- int getServerPort()：返回服务器接受此请求所用的端口号。
- BufferedReader getReader()：返回解码过的请求体。
- String getRemoteAddr()：返回发送此请求的客户端 IP 地址。
- String getRemoteHost()：返回发送此请求的客户端主机名。
- void setAttribute（String key,Object obj）：设置属性的属性值。
- String getRealPath（String path）：返回一虚拟路径的真实路径。
- getHeaders（String s）：获取头文件中指定头名字的全部值的一个枚举。
- getServerPort()：获取服务器的端口号。

上面讲解了这么多方法，读者肯定很难理解，下面通过一个实例来讲解上述方法的用法。

实例 28：向客户端返回信息

我们知道使用 Request 对象可以向客户端返回信息，下面通过几段代码讲解使用 Request 对象的流程，其代码（光盘：源代码/第 9 章/9-2/）如下。

编写文件 index.jsp，具体代码如下。

```
<%@page contentType="text/html;charset=gb2312"%>
<HTML>
      <HEAD>
            <TITLE>request 对象</TITLE>
      </HEAD>
      <BODY>
      <H3>计算两数之和</H3>
      <FORM METHOD=POST ACTION="common.jsp">
            数 1：<INPUT TYPE="text" NAME="num1"><BR>
            数 2：<INPUT TYPE="text" NAME="num2"><BR>
                <INPUT TYPE="submit" VALUE="加法">
      </FORM>
      </BODY>
</HTML>
```

新建文件 common.jsp，其代码如下。

```
<%@page contentType="text/html;charset=gb2312"%>
<HTML>
      <HEAD>
```

```
            <TITLE>request 对象</TITLE>
        </HEAD>
        <BODY>
        <%
            request.setCharacterEncoding("gb2312");
            String num1 = request.getParameter("num1");
            String num2 = request.getParameter("num2");
            int i = 0 ;
            int j = 0 ;
            try
            {
                i = Integer.parseInt(num1) ;
                j = Integer.parseInt(num2) ;
        %>
            两数之和是：<FONT COLOR="RED"><%=(i+j)%></FONT>
        <%
            }
            catch(Exception e)
            {
        %>
                Input Error!!!
        <%
            }
        %>
        </BODY>
    </HTML>
```

编码完毕，在浏览器运行后会得到如图 9-6 所示的效果。

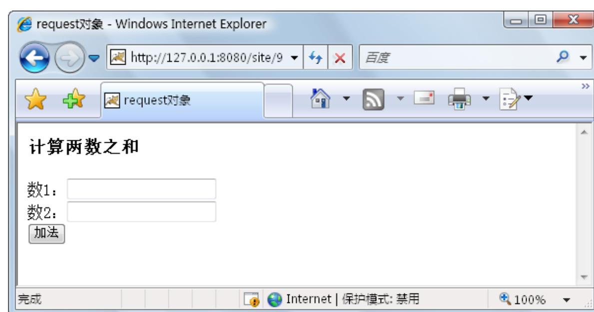


图 9-6　第一个页面

在表单中随意输入两个数，然后单击“加法”按钮后会得到如图 9-7 所示的效果。



图 9-7　request 对象



多学一招

在上面的实例代码中使用了 Request 对象，目的是用它来接受客户端的信息。下面通过一段代码继续讲解 Request 对象的用法，让大家更深层次地认识 Request 对象，从而达到灵活应用的效果。代码保存在（光盘：源代码/第 9 章/9-3/），实现流程如下。

新建一个文件 index.jsp，其代码如下。

```
<%@page contentType="text/html;charset=gb2312"%>
<HTML>
      <HEAD>
            <TITLE>request 对象</TITLE>
      </HEAD>
   <BODY>
      <FORM METHOD=POST ACTION="common.jsp">
            <SELECT NAME="city">
                  <OPTION VALUE="武汉">武汉</OPTION>
                  <OPTION VALUE="上海">上海</OPTION>
                  <OPTION VALUE="北京">北京</OPTION>
                  <OPTION VALUE="天津">天津</OPTION>
                  <OPTION VALUE="重庆">重庆</OPTION>
            </SELECT>
            <INPUT TYPE="submit" VALUE="提交">
      </FORM>
      </BODY>
</HTML>
```

新建文件 common.jsp，其代码如下。

```
<%@page contentType="text/html;charset=gb2312"%>
<HTML>
      <HEAD>
            <TITLE>request 对象</TITLE>
      </HEAD>
      <BODY>
      <%
            request.setCharacterEncoding("gb2312") ;
            String city = request.getParameter("city") ;
      %>
      <h1>城市：<%=city%></h1>
      </BODY>
</HTML>
```

编码完毕后，在浏览器中运行后会得到如图 9-8 所示的效果。

在表单中选择一个选项，然后单击“提交”按钮后会得到如图 9-9 所示的效果。

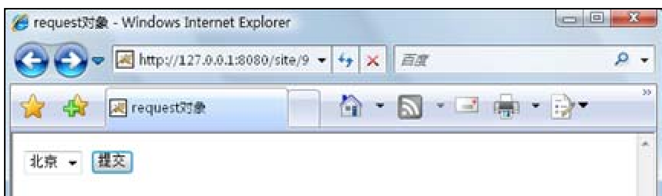


图 9-8　浏览后的效果

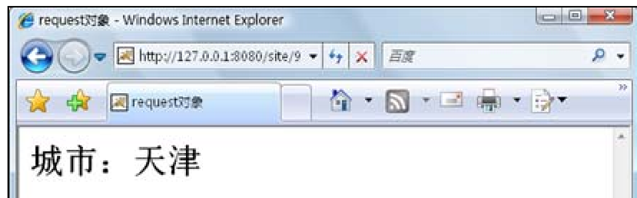


图 9-9　浏览的结果

9.5.2　对象 Response

Response 对象是 HttpServletResponse 类的对象，功能是当服务器返回一个信息时封装所返回的信息，然后将其返回到客户端以响应请求。在 Response 内置对象中提供了一些功能强大的方法，各个方法的具体说明如下。

- String getCharacterEncoding()：返回响应用的是何种字符编码。
- ServletOutputStream getOutputStream()：返回响应的一个二进制输出流。
- PrintWriter getWriter()：返回可以向客户端输出字符的一个对象。
- void setContentLength（int len）：设置响应头长度。
- void setContentType（String type）：设置响应的 MIME 类型。
- sendRedirect（java.lang.String location）：重新定向客户端的请求。

实例 29：使用对象 Response

本实例的目的是讲解使用对象 Response 的流程，其代码保存在（光盘：源代码/第 9 章/9-4/）。本实例的具体实现流程如下。

新建文件 index.html，其代码如下。

```
<%@
  page contentType="text/html;charset=gb2312"
 %>
 <html>
    <head>
          <title>首页</title>
    </head>
    <body>
          <form action="usingResponse.jsp" method="post">
             <input type="radio" name="using" value="a">刷新网页<br>
             <input type="radio" name="using" value="b">缓冲区操作<br>
             <input type="submit" value="提交">
```

```
            </form>
        </body>
    </html>
```

新建文件 usingResponse.jsp，其代码如下。

```
    <%@
      page contentType="text/html;charset=gb2312"
    %>
    <html>
        <head>
            <title>处理 response</title>
        </head>
        <body>
             <%
              String using =request.getParameter("using");
              if(using.equals("a"))
              response.sendRedirect("show1.jsp");
              else if(using.equals("b"))
              response.sendRedirect("show2.jsp");
              %>
        </body>
    </html>
```

新建文件 show1.jsp，其代码如下。

```
    <%@
      page contentType="text/html;charset=gb2312"
    %>
    <html>
        <head>
            <title></title>
        </head>
        <body>
            <%!int c=0;%>
            <%
             if(c==4)
             c=0;
             response.setHeader("Refresh","1");
             c++;
             %>
             <img src="<%=c%>.jpg"><br>
        </body>
    </html>
```

新建文件 show2.jsp，其代码如下。

```
    <%@ page contentType="text/html;charset=gb2312"
```

```
    %>
    <html>
        <head>
            <title></title>
        </head>
        <body>
            <%
            out.println("缓冲区大小:");
            out.println(response.getBufferSize()+"<br>");
            out.println("是否写出缓冲区的数据:");
            out.println(response.isCommitted()+"<br>");
            %>
        </body>
    </html>
```

运行上述程序，选择第一个单选按钮选项“刷新网页”，如图 9-10 所示。

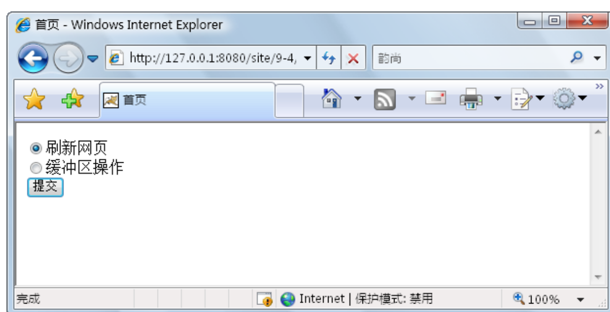


图 9-10　选择选项

单击“提交”按钮后会得到如图 9-11 所示的效果。

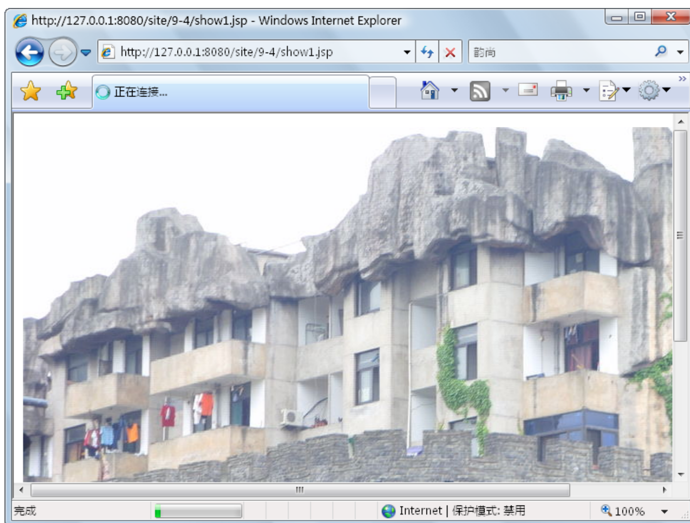


图 9-11　不停更换图片

如果选择第二个单选按钮选项“缓冲区操作”，单击“提交”按钮后会得到如图 9-12 所示的效果。

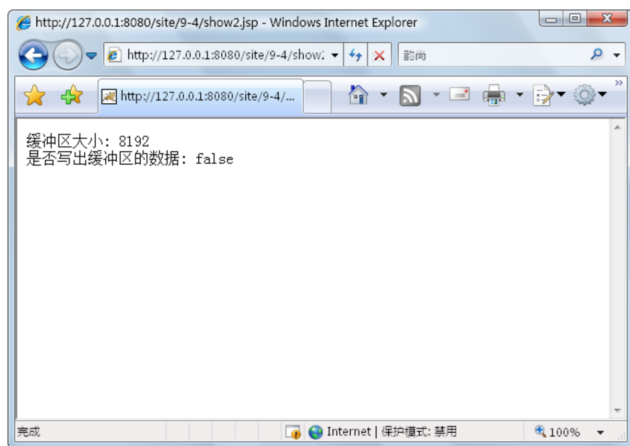


图 9-12　缓冲区操作



多学一招

在上面的实例代码中，讲解了使用 Response 对象的流程。下面将通过一段代码，让大家多学习一招使用 Response 对象的方法，其代码（光盘：源代码/第 9 章/9-5.jsp）如下。

```
<%@ page contentType="text/html; charset=gb2312" language="java"
import="java.sql.*" errorPage="" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>request 进行缓冲</title>
</head>
<body>
<%
out.print("10 秒钟进行刷新一次");
response.setHeader("Refresh","10");
%>
</body>
</html>
```

运行上述程序，得到如图 9-13 所示的效果。



图 9-13　刷新

9.5.3　Session 对象

Session 是指一个终端用户与交互系统进行通信的时间间隔，通常指从注册进入系统到注销退出系统之间所经过的时间。在 JSP 中，session 的会话是 HttpSession 类的实例，其常用方法如下。

- public void setAttribute（String Key,Object obj）：Session 对象类似于散列表，可以调用该方法将参数 Object 指定的对象 obj 添加到 Session 对象中，并为其添加的对象指定一个索引关键字。如果添加的两个对象的关键字相同，则先前添加的对象被清除。
- public Object getAttribute（String Key）：获取 Session 对象中含有的关键字是 Key 的对象，由于任何对象都可以添加到 Session 对象中，所以在该方法取回对象时应进行强制类型转换。
- public Enumeration getAttributeName()：Session 对象调用该方法产生一个枚举对象，该枚举对象使用 nextElement()方法遍历 Session 对象所含有的全部对象。
- public long getCreationTime()：Session 对象调用该方法用以获取对象创建的时间，单位是 ms，从 1970 年 7 月 1 日午夜起至该对象创建时刻所走过的毫秒数。
- public long getLastAccessedTime()：获取当前 Session 对象最后一次被操作的时间。
- public int getMaxInactiveInterval()：得到允许 Session 对象处于非活动状态的最长时间。
- public vodi setMaxInactiveInterval(int n)：设置允许 Session 对象处于非活动状态的最长时间，单位是 s。
- public void removeAttribute(String Key)：从当前 Session 对象中删除关键字是 Key 的对象。
- public String getId()：获取 Session 对象的编号。
- invalidate()：使 Session 对象无效。
- long getCreationTime()：返回 Session 对象的创建时间。
- public String getId()：返回 Session 创建时 JSP 引擎为它设置的唯一 ID 号。
- long getLastAccessedTime()：返回此 Session 中客户端最近一次的请求时间。
- int getMaxInactiveInterval()：返回两次请求间隔多长时间此 Session 被取消（ms）。
- String[] getValueNames()：返回一个包含此 Session 中所有可用属性的数组。
- boolean isNew()：返回服务器创建的一个 Session，判断客户端是否已经加入。
- void setMaxInactiveInterval()：设置两次请求间隔多长时间此 Session 被取消（ms）。

接下来将通过几段代码来讲解使用 Session 对象的流程，其代码保存在（光盘：源代码/第 9 章/9-5/），具体实现流程如下。

新建文件 index.jsp，其代码如下。

```
<%@ page contentType="text/html; charset=gb2312" language="java"
import="java.sql.*" errorPage="" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
```

```
<title>session 对象的应用</title>
</head>
<script language="javascript" type="">
function Mycheck(){
if (form.account.value=="")
{ alert("请输入用户名！");form.account.focus();return  false;}
if (form.password.value=="")
{ alert("请输入用户登录密码！");form.password.focus();return false;}
form.submit();
}
</script>
<body>
<table width="334" height="171" border="0" align="center" cellpadding
="0" cellspacing="0">
  <tr>
    <td>
     <br>
      <form name="form" method="post" action="dealwith.jsp?method=0"
onSubmit="return Mycheck()">
    <table width="250" border="0" align="center">
      <tr>
        <td width="62" height="20">用户名:</td>
        <td width="172">
          <input name="account" type="text" id="account">
       </td>
      </tr>
      <tr>
        <td height="20">密  码：</td>
        <td><input name="password" type="password" id="password"></td>
      </tr>
      <tr>
        <td height="20" colspan="2"><div align="center">
  <input type="submit" name="Submit" value="登录">
  
          <input type="reset" name="Submit2" value="重置">
        </div></td>
        </tr>
    </table>
    </form>
         </td>
  </tr>
</table>
</body>
</html>
```

新建文件 frward.jsp，其代码如下。

```
<%@ page contentType="text/html; charset=gb2312" language="java"
import="java.sql.*" errorPage="" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>session 对象的应用</title>
</head>
<body>
<table width="334" height="171" border="0" align="center" cellpadding
="0" cellspacing="0">
  <tr>
    <td>
         <br>
   <table width="250" border="0" align="center">
      <tr>
        <td width="62" height="20">用户名:</td>
        <td width="172"><%=session.getAttribute("account")%></td>
      </tr>
      <tr>
        <td height="20">密  码：</td>
        <td><%=session.getAttribute("password")%></td>
      </tr>
    </table>
   </td>
  </tr>
</table>
<table width="328" height="42" border="0" align="center">
  <tr>
    <td><div align="center"><a href="dealwith.jsp?method=1">安全退出</a>
</div></td>
  </tr>
</table>
</body>
</html>
```

新建文件 dealwith.jsp，其代码如下。

```
<%@ page contentType="text/html; charset=gb2312" language="java"
import="java.sql.*" errorPage="" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>session 对象的应用</title>
</head>
<body>
<%
int method=Integer.parseInt(request.getParameter("method"));
if(method==0){
String account=request.getParameter("account");
```

```
    account = new String(account.getBytes("ISO8859_1"), "GBK");
    session.setAttribute("account",account);
    String password=request.getParameter("password");
    password = new String(password.getBytes("ISO8859_1"), "GBK");
    session.setAttribute("password",password);
    response.sendRedirect("forward.jsp");
    }
    if(method==1){
    session.invalidate();
    response.sendRedirect("index.jsp");
    }
    %>
    </body>
    </html>
```

运行上述程序，得到如图 9-14 所示的效果。

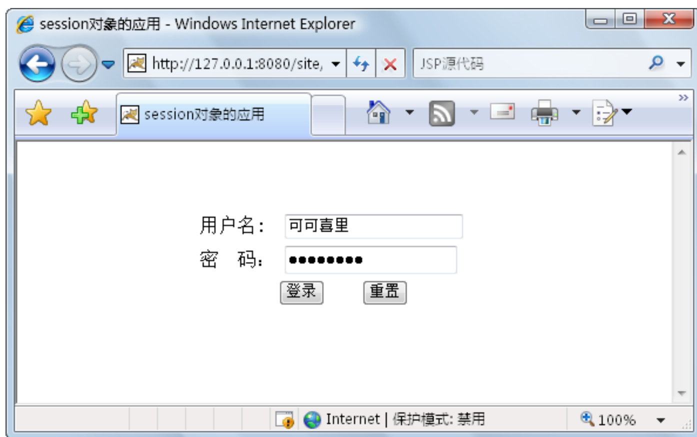


图 9-14　执行后的结果

输入相关信息后单击“登录”按钮，打开如图 9-15 所示的信息。

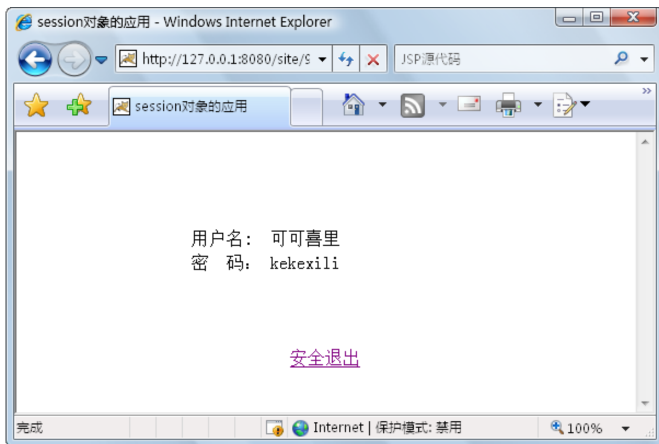


图 9-15　Session 对象

9.5.4　Out 对象

Out 对象是 JSPWriter 类的实例，是向客户端输出内容的常用对象。Out 对象的功能是将数据输出到客户端的浏览器上，在里面提供了一系列用来控制输出的方法，各个方法的具体说明如下。

- ❑ void clear()：清除缓冲区的内容。
- ❑ void clearBuffer()：清除缓冲区的当前内容。
- ❑ void flush()：清空流。
- ❑ int getBufferSize()：返回缓冲区大小，如果不设置设缓冲区则为 0。
- ❑ int getRemaining()：返回缓冲区还剩余多少可用空间。
- ❑ boolean isAutoFlush()：返回缓冲区满时，设置是自动清空还是抛出异常。
- ❑ void close()：关闭输出流。

下面通过一段代码来讲解使用 Out 对象的流程，其代码（光盘：源代码/第 9 章/9-6/）如下。

```
<%@ page
 contentType="text/html;charset=GBK"
%>
 <html>
     <head>
         <title>out 对象的使用</title>
     </head>
     <body>
    <%
            out.println("这是一个 out 对象<br>");
            out.clearBuffer();
            out.println("欢迎你访问<br>");
            out.flush();
            out.println("日期<br>");
            out.println(new java.util.Date().toLocaleString()+"<br>");
            out.println("剩余缓冲区大小"+out.getRemaining()+"<br>");
            out.println("缓冲区大小"+out.getBufferSize()+"<br>");
            out.println("AutoFlush"+out.isAutoFlush());
     %>
                </body>
 </html>
```

运行上述程序后得到如图 9-16 所示的效果。

9.5.5　Application 对象

Application 对象实现了用户间数据的共享，可存放全局变量。Application 开始于服务器的启动，直到服务器关闭时结束。在此期间，Application 对象将一直存在。这样在用户的前后连接或不同用户之间的连接中，可以对此对象的同一属性进行操作。在任何地方对此对象

属性的操作，都将影响到其他用户对此的访问。服务器的启动和关闭决定了 Application 对象的生命。Application 对象是 ServletContext 类的实例，包括如下常用的方法。

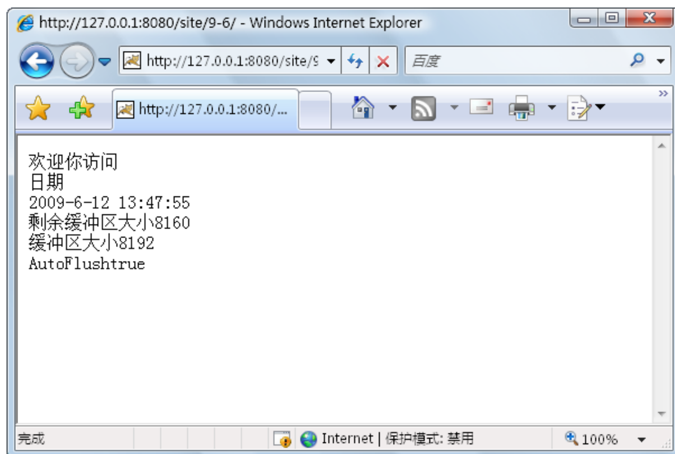


图 9-16　Out 对象

- Object getAttribute（String name）：返回给定名的属性值。
- Enumeration getAttributeNames()：返回所有可用属性名的枚举。
- void setAttribute（String name,Object obj）：设定属性的属性值。
- void removeAttribute（String name）：删除一个属性及其属性值。
- String getServerInfo()：返回 JSP（SERVLET）引擎名及版本号。
- String getRealPath（String path）：返回一个虚拟路径的真实路径。
- ServletContext getContext（String uripath）：返回指定 WebApplication 的 application 对象。
- int getMajorVersion()：返回服务器支持的 Servlet API 的主版本号。
- int getMinorVersion()：返回服务器支持的 Servlet API 的次版本号。
- String getMimeType（String file）：返回指定文件的 MIME 类型。
- URL getResource（String path）：返回指定资源(文件及目录)的 URL 路径。
- InputStream getResourceAsStream（String path）：返回指定资源的输入流。
- RequestDispatcher getRequestDispatcher（String uripath）：返回指定资源的 Request Dispatcher 对象。
- Servlet getServlet（String name）：返回指定名的 Servlet。
- Enumeration getServlets()：返回在这个 Servelet 环境下所有已知的 Servlet 对象的列表。这个列表总是包含这个 Servlet 自身。
- Enumeration getServletNames()：用来返回一个 String 对象的列表，该列表表示了在这个 Servlet 环境下所有已知的 Servlet 对象名。这个列表总是包含这个 Servlet 自身。
- void log（String msg）：把指定消息写入 Servlet 的日志文件。
- void log（Exception exception,String msg）：把指定异常的栈轨迹和错误消息写入 Servlet 的日志文件。

- void log（String msg,Throwable throwable）：把栈轨迹和给出的 Throwable 异常的说明信息写入到 Servlet 的日志文件。

接下来将通过几段代码来讲解使用 Application 对象的流程，其代码保存在（光盘：源代码/第 9 章/9-7/）中，具体实现流程如下。

新建文件 liuyan1.jsp，其代码如下。

```
<%
  String  temp=new  String();
  temp=(String)application.getAttribute("num");
  if(temp==null){
  application.setAttribute("num","0");
  }
  %>
  <html>
  <head>
  <title>留言板</title>
  <meta http-equiv="Content-Type" content="text/html; charset=gb2312">
  <script language="JavaScript">
  <!--
  function MM_openBrWindow(theURL,winName,features)  {  //v2.0
     window.open(theURL,winName,features);
  }
  //-->
  </script>
  </head>

  <body bgcolor="#FFFFFF" text="#000000">
  <p> </p>
  <p align="center"><b><font size="5"color="#FF6633">留言板</font></b>
</p>
  <hr  size="1">
  <form  name="form1"  method="post"  action="liuyan2.jsp">
     <table width="60%" border="0" cellspacing="1" cellpadding="3"
align="center" bgcolor="#999999">
        <tr bgcolor="#FFFFFF">
           <td width="20%">留言作者：</td>
           <td width="80%">
              <input type="text" name="txtaut">
           </td>
        </tr>
        <tr bgcolor="#FFFFFF">
           <td width="20%">留言标题：</td>
           <td width="80%">
              <input type="text" name="txttit">
           </td>
        </tr>
```

```
            <tr bgcolor="#FFFFFF">
                <td width="20%">留言内容：</td>
                <td width="80%"> </td>
            </tr>
            <tr bgcolor="#FFFFFF">
                <td width="20%"> </td>
                <td width="80%">
                    <textarea name="txtart" cols="45" rows="7"></textarea>
                </td>
            </tr>
            <tr bgcolor="#FFFFFF">
                <td width="20%"> </td>
                <td width="80%"> </td>
            </tr>
            <tr bgcolor="#FFFFFF">
                <td colspan="2">
                    <div align="center">
                       <input type="submit" name="cmdok" value="添加到留言薄">
                        <input type="reset" name="cmdreset" value="重新填写留言">
                        <input type="button" name="cmdread" value="查看所有留言"
        onClick="MM_openBrWindow('liuyan3.jsp'),'','toolbar=no,
  location=no,status=no,menubar=no,scrollbars=no,width=400,height=400'">
                    </div>
                </td>
            </tr>
        </table>
    </form>
    <p> </p>
    </body>
    </html>
```

新建文件 liuyan2.jsp，其代码如下。

```
  <html>
    <head>
    <title>留言板</title>
    <meta http-equiv="Content-Type" content="text/html; charset=gb2312">
    <script language="JavaScript">
    <!--
    function MM_openBrWindow(theURL,winName,features) { //v2.0
        window.open(theURL,winName,features);
    }
    //-->
    </script>
    </head>

    <body bgcolor="#FFFFFF" text="#000000">
```

```
  <p> </p>
  <p align="center"><b><font size="5" color="#FF6633">留　言　板
</font></b></p>
  <hr size="1">
  <%
  int n;
  String temp=new String();
  String temp1=new String();
  String temp2=new String();
  String temp3=new String();
  temp1=request.getParameter("txttit");
  temp2=request.getParameter("txtaut");
  temp3=request.getParameter("txtart");
  n=temp1.length()*temp2.length()*temp3.length();
  if(n!=0){
  temp=(String)application.getAttribute("num");
  n=Integer.parseInt(temp);
  n=n+1;
  temp=temp.valueOf(n);
  application.setAttribute("num",temp);
  application.setAttribute("tit"+temp,temp1);
  application.setAttribute("aut"+temp,temp2);
  application.setAttribute("art"+temp,temp3);
  %>
  <p> </p>
  <p align="center">留言成功！</p>
  <%
  }
  else{
  %>
  <p align="center"><font  color="#FF0000">不添加作者、标题和内容，留言失
败！</font></p>
  <%
  }
  %>
  <p align="center"><a  href="liuyan1.jsp">返回首页</a>  </p>
  </body>
  </html>
```

新建文件 liuyan3.jsp，其代码如下。

```
<html>
  <head>
  <title>留言板</title>
  <meta http-equiv="Content-Type" content="text/html; charset=gb2312">
  </head>
```

```
<body bgcolor="#FFFFFF" text="#000000">
<p> </p>
<p align="center"><b><font size="5" color="#FF6633">留  言  板
</font></b></p>
<hr size="1">
<%
int n;
String temp=new String();
String temp1=new String();
String temp2=new String();
String temp3=new String();
temp=(String)application.getAttribute("num");
n=Integer.parseInt(temp);
if(n==0){
%>
<p> </p>
<p align="center">目前还没有文章! </p>
<%
}
else{
%>
<table width="60%" border="1" cellspacing="0" cellpadding="5" align=
"center"  bordercolor="#999999">
    <%
    int i;
    for(i=1;i<=n;i++){
    temp=temp.valueOf(i);
    temp1=(String)application.getAttribute("tit"+temp);
    temp2=(String)application.getAttribute("aut"+temp);
    temp3=(String)application.getAttribute("art"+temp);
    %>
    <tr>
        <td bgcolor="#CCFFCC" height="27"><b><%=temp%>.标题：<%=
temp1%>作者：<%=temp2%></b></td>
    </tr>
    <tr>
        <td><%=temp3%></td>
    </tr>
    <%
    }
}
    %>
</table>
<p align="center"> </p>
</body>
</html>
```

在浏览器输入网站地址后将显示留言信息，如图 9-17 所示。

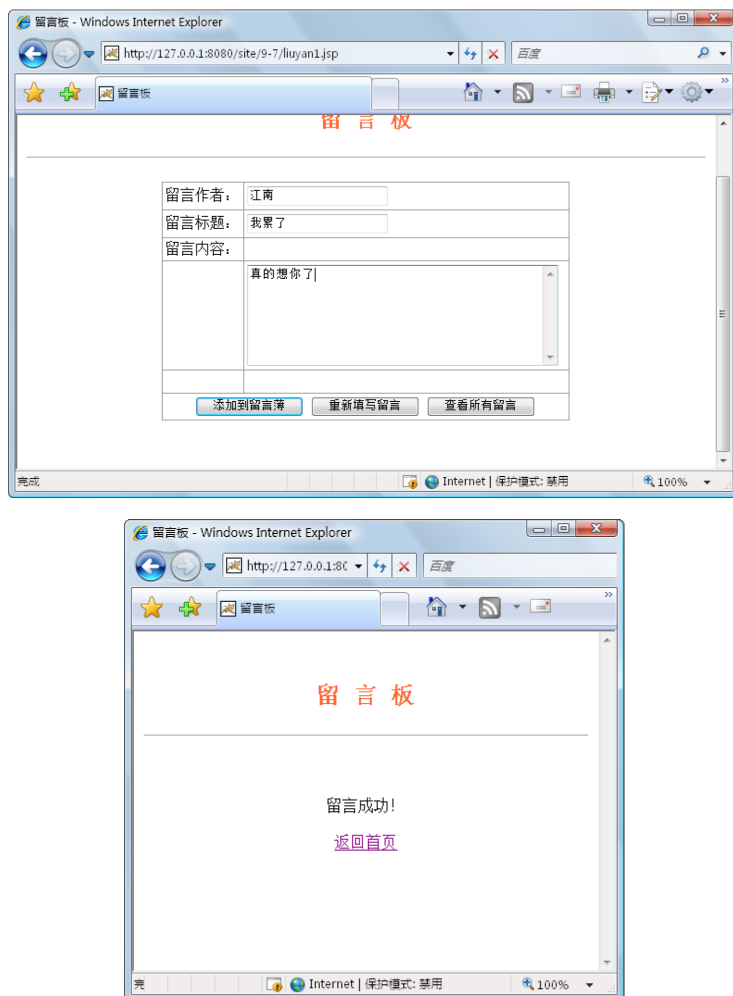


图 9-17　留言

当用户发布留言后，可以查看已经发布的留言信息，如图 9-18 所示。

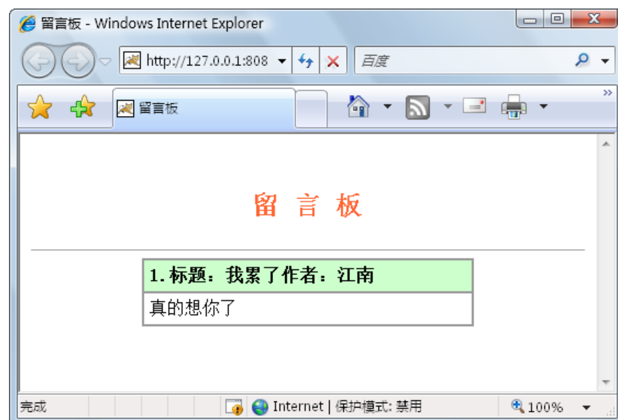


图 9-18　查看自己的留言

9.6 疑难问题解析

本章详细讲解了 JSP 指令和内置对象的基本知识。本节中，将对本章中比较难以理解的问题进行讲解。

读者疑问：在本章中，出现了几个词，JSP、Java Web，它们之间有什么关系，它们又表示着什么？

解答：JSP 是 Sun 公司推出的一门网络编程语言，主要是标记性的语言；Java Web 指的是 Java 在 Web 领域实现的一系列开发，这是 Java 最擅长的，JSP 不是 Java Web，Java Web 包含着 JSP，没有 JSP，Java Web 很难开发。

读者疑问：JSP 语言是不是十分简单？学习它需要注意什么？

解答：JSP 是最简单的脚本语言，用户只需要记住它的标记就可以用了，它没有什么技巧，Java Web 发挥重要功能的时候，还得依靠 Java，JSP 是简单的，这一章只进行了简单的讲解，下一章还会继续学 JSP 的相关知识。



职场点拨——谈试用期

无论是一名应届毕业生，还是一名资深程序员，只要是来到一家新的公司上班，必须首先面对试用期。试用期就是检验时期，是进入职场后的第一阶段。如果表现不够好，公司不会留用。在试用期间，究竟怎样才能获得公司的认可呢？以下 3 条建议值得借鉴。

（1）谦虚谨慎，小中见大

作为一名新人，特别是刚出校门的大学生，往往志向都比较远大，恨不得一下子就做成一件大事，博得领导的赏识。然而往往因为缺乏经验，所以做的工作并不被领导认可。在实际工作中，领导可能会对你的工作进行指导，有时也会提出善意的批评，这就要求我们能够正确认识，虚心接受别人的意见，不要以为自己的文凭比别人高就自以为是。只有这样，才能学到职场经验。

（2）幽默风趣，富有魅力

无论是学习环境还是工作环境，都讲究有一个活跃的气氛。在气氛融洽而又活跃的环境中工作，人的心情较为愉悦，办事效率也会高些。无论是在工作中还是在生活中，大家都应互相鼓励并共同进步，在麻烦和矛盾面前，不斤斤计较。在试用期要用幽默诙谐的语言和恰如其分的措词来激活工作的氛围，给领导和同事留下好的印象。

（3）关键时刻显身手

在试用期，有关领导往往会给毕业生布置一些任务，有些甚至是难度较大的任务，借此来考查你的能力。都说好钢用在刀刃上，在完成这些任务时，一定要把这些任务办得圆圆满满、万无一失。

第 10 章　标签和动作指令

在上一章内容中，相信读者朋友已经熟悉了 JSP 中基本标签的知识。本章将继续学习 Java Web 的一些基础语法，从而更全面地掌握 Java Web 知识；将向读者讲解 Java Web 动作指令和 JSP 2.0 的一些新特性。通过本章能学到如下知识。

- 动作指令。
- 定义标签。
- JSP 2.0 的新特性。
- 职场点拨——同事间的距离。

小菜已经工作 1 年，职位程序员。

2010 年 XX 月 XX 日，多云

不知不觉间已经工作 1 年了，每天大部分时间都在办公室里，周围是我亲爱的同事。我愈发感觉同事关系很重要，暗下决心一定要和他们好好地相处……



一问一答

小菜：“同事相处真是一门学问，我一定要和他们好好地相处。”

Wisdom：“呵呵，只要你在职场中，就需要面对同事关系。同事不但能在技术上给你点拨，而且能帮你成长。即使是在激烈的辩论中，也能使你意识到问题的所在，提高你的业务能力。所以你要好好处理同事关系，能学习到不少的生活知识。”

小菜：“嗯，言归正传，本章我们学习的标签和动作指令好学吗？”

Wisdom：“在本章的内容中，重点和难点是 Java 的动作指令，动作指令只是运行时的脚本动作，这个动作能辅助实现某个具体的功能。”

10.1　认识标签和指令

在本章的内容中，重点和难点是 Java 的动作指令，下面通过一段代码让读者品味动作指令的功能，其代码保存在（光盘：源代码/第 10 章/10-1/）中，具体实现流程如下。

新建一个文件 index.jsp，其代码如下。

```
<%@
  page contentType="text/html; charset=gb2312"
```

```
%>
<html>
    <html>
        <title></title>
    </html>
    <body>
         <jsp:plugin type="applet" code="image.class"/>
    </body>
</html>
```

新建一个文件 image.java，其代码如下。

```
import java.awt.*;
import java.applet.Applet;
public class image extends Applet
{
   Image person;

   public void init()
   {
        person=getImage(getDocumentBase(),"122.jpg");
   }

   public void paint(Graphics g)
   {
        g.drawImage(person,100,100,this);
   }
}
```

在上述代码中，需要预先准备一张 jpg 图片，然后将其命名为 122.jpg。首先编译文件 image.java，让它产生.Class 文件，随后在浏览器中输入浏览地址，运行后的效果如图 10-1 所示。

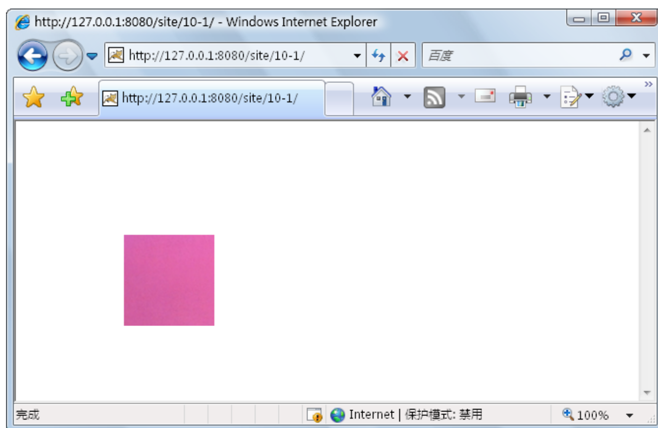


图 10-1　编译后的结果

10.2　7 个动作指令

动作指令与前面讲解的编译指令有所不同，编译指令通常会结合后面讲解的 Servlet 引擎来处理信息，而动作指令只是运行时的脚本动作。在 Java Web 中，一共有 7 个动作指令，它们在 Java Web 中的作用不言而喻。下面将详细讲解这 7 个动作指令的基本知识。

10.2.1　forward 动作指令

forward 指令用于将页面响应控制转发给另外的页面。在转发时，既可以转发给静态的 HTML 页面，也可以转发到动态的 JSP 页面，或者转发到容器中的 Servlet。JSP 1.0 中 forward 指令的格式如下。

```
<jsp: forward page=" {relativeURLI<%=expression%>}"/>
```

JSP 1.1 中 forward 指令的格式如下。

```
<jsp:forward page="url"/>
<jsp:param name="fname" value="fvalue"/>
</jsp:forward/>
```

上述第二种格式能够在转发时增加额外的请求参数，增加的请求参数的值可以通过 HttpServlet 中 Request 类的 getParameter 方法获取。

各个参数的介绍如下。

- page：一个相对的 URL 路径。
- fname：属性是参数名。
- fvalue：参数值。

实例 30：使用 forward 指令

forward 是一个十分重要的指令，下面通过一段代码来演示使用 forward 指令的流程，其代码保存在（光盘：源代码/第 10 章/10-2/）中。

新建一个文件 example.html，其页面代码如下。

```
<%@ page contentType="text/html; charset=gb2312" %>
<html>
    <head>
        <title>
</title>
    </head>
    <body>
        <h1>其实很简单，真的，相信我
</h1>
    </body>
</html>
```

新建一个文件 index.jsp，其页面代码如下。

```
<%@ page contentType="text/html; charset=gb2312" %>
```

```
<html>
    <head>
        <title>
forward 动作</title>
    </head>
    <body>
        <jsp:forward page="example.html"/>
    </body>
</html>
```

上面的代码十分简单，当编码完毕后，在浏览器运行后会得到如图 10-2 所示的效果。

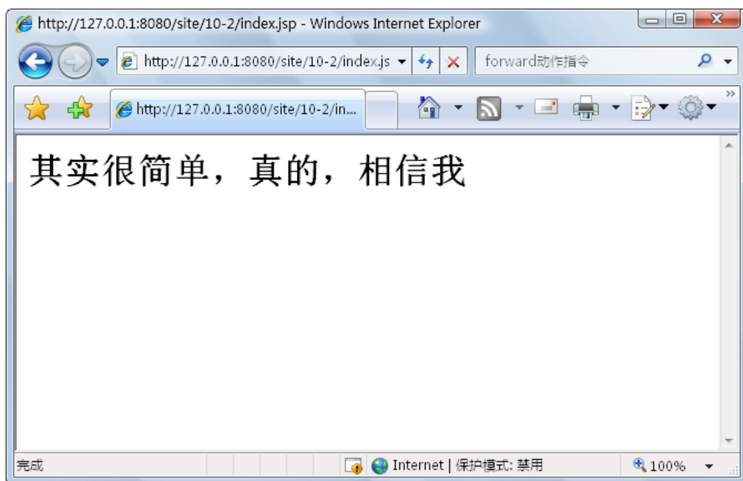


图 10-2　forward 指令


多学一招

从图 10-2 所示的执行效果可看出，可以在转发的新页面中访问到 forward 指令增加的参数，这样就可以将 forward 用在表单中，实现一些功能。当 jsp:forward 指令转发请求时，客户端的请求参数不会丢失。例如，在下面的表单提交页面代码中，该页面没有任何动态的内容，是一个静态的 HTML 页面，仅将表单域的值提交到 jsp-forward 页，其代码保存在（光盘：源代码/第 10 章/10-3/）中。

新建一个文件 index.html，其页面代码如下。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>提交</TITLE>
</HEAD>
<BODY>
<!-- 表单提交页面 -->
<form id="login" method="post" action="jsp-forward.jsp">
```

```
<INPUT TYPE="text" NAME="username">
<INPUT TYPE="submit" value="login">
</FORM>
</BODY>
</HTML>
```

新建一个 forward-result.jsp 页面，其代码如下。

```
<%@ page contentType="text/html; charset=gb2312" language="java" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>forward 结果页</TITLE>
</HEAD>
<BODY>
<!-- 使用 request 内置对象获取 age 参数的值 -->
<%=request.getParameter("age")%>
<!-- 输出 username 请求参数的值 -->
<%=request.getParameter("username")%>
</BODY>
</HTML>
```

上面的代码十分简单，编码完毕后，在浏览器中运行后会得到如图 10-3 所示的效果。

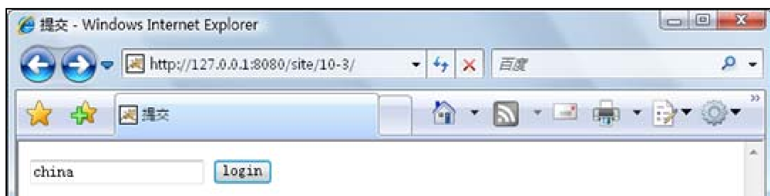


图 10-3　输入的内容

输入内容后，然后将其提交，得到如图 10-4 所示的页面。

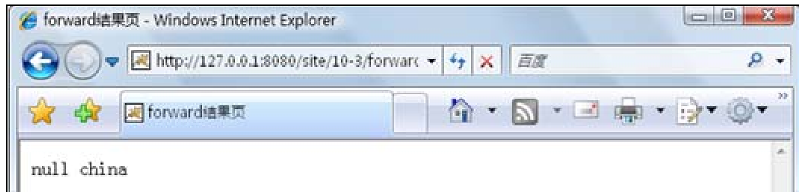


图 10-4　提交后的结果

10.2.2　param 动作指令

param 动作指令有 3 个功能，具体说明如下。

1）当与 include 指令结合使用时，param 指令能够将参数值传入被导入的页面。

2）当与 forward 指令结合使用时，param 指令能够将参数值传入被转向的页面。

3）当与 plugin 指令结合使用时，则用于将参数传入页面中的 JavaBean 实例或 Applet 实例。

param 指令的语法格式如下。

```
<jsp:param name="fname" value="fvalue"/>
```

参数介绍如下。

- ❑ name：传递参数的名称。
- ❑ value：表示传递参数的值。

提示：param 指令用于设置参数值，这个指令本身不能单独使用，因为单独的 param 指令没有实际意义。

10.2.3　include 动作指令

include 指令是一个动态指令，也能够导入某个页面，在导入时会每次检查被导入页面的改变。下面是 include 指令的语法格式：

```
<jsp:include page=”fiel” flush=”true”/>
```

其中，file 为加载文件所在的位置，相对路径；flush 属性设置为 true，如果设置为 false，将不能自动刷新。若在加载文件或网页时需要传递参数，那么使用语法如下。

```
<jsp:include page="fiel" flush="true"/>
<jsp:param name="fname" value="fvalue"/>
</jsp:include>
```

参数介绍。

flush 属性用于指定输出缓存是否转移到被导入文件中。如果指定为 true，则包含在被导入文件中；如果指定为 false，则包含在原文件中。对于 JSP 1.1，只能设置为 false。对于第二种格式，可用于在导入页面中加入参数值。

下面通过一段代码来讲解使用 include 指令的流程，新建一个 user.html 页面，其代码（光盘：源代码/第 10 章/10-4/）如下。

```
<%@
   page contentType="text/html;charset=gb2312"
 %>
<html>
    <head>
        <title>include 动作</title>
    </head>

    <body>
         <h3>欢迎你</h3>
        <h3>这是一个登录界面</h3>
        <h3>请登录</h3>
```

```
            <form action="usinginclude.jsp" name="myform" method="post">
             输入你的昵称:<input type="text" name="name"><br>
             输入你的密码:<input type="text" name="password"><br>
                   <input type="submit" value="确定">
              </form>
        </body>
</html>
```

新建一个 index.jsp 文件，其代码如下。

```
<%@
   page contentType="text/html;charset=gb2312"
 %>
<%@
  include file="user.html"
 %>
 <html>
     <head>
         <title></title>
     </head>
     <body>
     </body>
 </html>
```

上面的代码十分简单，编码完毕后，在浏览器中运行后会得到如图 10-5 所示的效果。

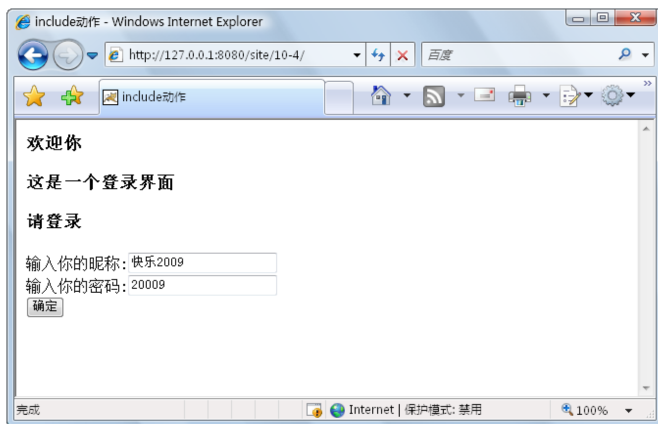


图 10-5　include 动作

10.2.4　plugin 动作指令

plugin 动作指令主要用于下载服务器端的 JavaBean 或 Applet 到客户端执行。由于程序在客户端执行，因此客户端必须安装虚拟机。其语法格式如下。

```
  <jsp:plugin type=bean applet"
  code="classFileName"codebase="classFileDiretoryName"
```

```
    [name="instanceName"][archive="URLtoArchive"]
    tal工gn="bottom 1topImiddle 1left 1right"]
    [heigh="displayPixels"][width="displayPixels"]
    [hspace="leftRightPixels"]
    [vspace="topBottomPiexels"]
    [jreversion=JREVersionNumber11.2"]
    [nspluginurl="URLToPlugin"]
    [iepluginurl="URLToPlugin"]>[<jsp:parames>
    [jsp:param name="parameterName" value="parameterValue"I>]
    </jsp:params>][<jsp:fallback>text message for user that can no see
  the plugin
  </jsp:fallback> ]
    </jsp:plugin>
```

参数介绍如下。

- ❑ type：指定被执行的 Java 程序的类型。
- ❑ code：指定被执行的文件名，该属性值必须以“class”扩展名结尾。
- ❑ codebase：指定被执行文件所在的目录。
- ❑ name：给该程序起一个名字用来标识该程序。
- ❑ archive：指向一些要预先载入的、将要使用到的类的路径。
- ❑ hspace，vspace：显示左右、上下的留白。
- ❑ jreversion：能正确运行该程序必需的 JRE 版本，默认值是 1.2。
- ❑ nspluginurl、iepluginurl：浏览器下载运行需要的 JRE 的版本。
- ❑ <JSP:fallback>：当不正确显示该 Applet 时，显示该指令中的文本提示。

下面通过几段代码来讲解使用 plugin 动作指令的流程，其代码保存在（光盘：源代码/第 10 章/10-5/）中。

新建一个 hello.java 文件，其代码如下。

```
import java.applet.*;
import java.awt.*;
//Applet继承java.applet.Applet类
public class Hello extends Applet
{
   //重写paint方法，该方法将在Applet上绘图
   public void paint(Graphics g)
   {
         //先绘出字符串，字符串通过getParameter方法获取
         g.drawString(getParameter("hello"),40,40);
         //设置颜色
         g.setColor(new Color(152,20,2));
         //画出一个矩形
         g.fillRect(50,60,200,150);
   }
}
```

将上述代码编译后形成 class 文件，然后新建一个 index.jsp 文件，其代码如下。

```
<%@ page contentType="text/html; charset=gb2312" language="java" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>plugin 就这么简单</TITLE>
</HEAD>
<BODY>
<!-- 使用 Applet -->
<jsp:plugin type="applet" code="Hello.class">
   <jsp:params>
        <jsp:param name="hello" value="我的世界我做主"/>
   </jsp:params>
   <jsp:fallback>
        <p>不能下载 jre 插件</p>
   </jsp:fallback>
</jsp:plugin>
</BODY>
</HTML>
```

上面的代码十分简单。编码完毕后，在浏览器中运行后会得到如图 10-6 所示的效果。

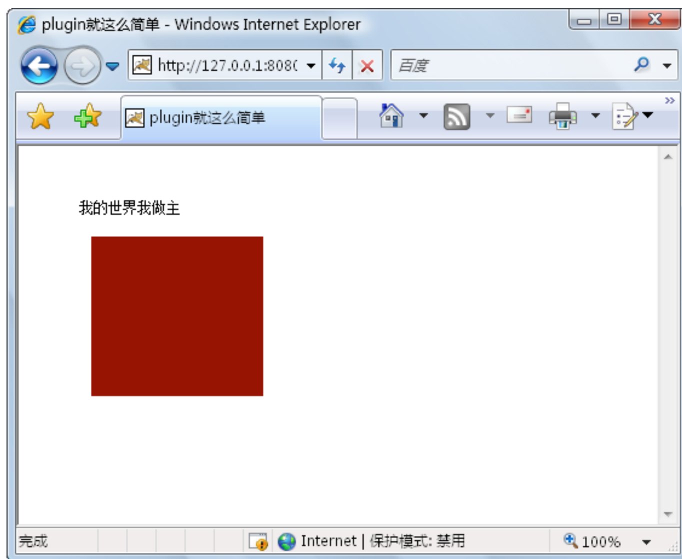


图 10-6　plugin 动作指令

10.2.5　useBean,setProperty,getProperty 动作指令

这 3 个指令都是和 JavaBean 密切相关的内容，其中 useBean 指令用于在 JSP 页面中初始化一个 Java 实例；setProperty 指令用于修改 JavaBean 实例的属性；getProperty 指令用于获取 JavaBean 实例的属性。

useBean 指令的语法格式如下。

```
<jsp:useBean id="name" class="classname" scope="pageIrequestIsessionI
application"/>
```

其参数介绍如下。

1）id 属性：是 JavaBean 的实例名。

2）class 属性：用于确定 JavaBean 的实现类。

3）scope 属性：用于确定 JavaBean 实例的生存范围，该范围有以下 4 个值。

- page：该 JavaBean 实例仅在该页面有效。
- request：该 JavaBean 实例在本次请求有效。
- session：该 JavaBean 实例在本次 session 内有效。
- application：该 JavaBean 实例在本应用内一直有效。

setProperty 指令的语法格式如下。

```
<jsp:setProperty name="BeanName" proterty="propertyName" value=
"value"/>
```

其参数介绍如下。

- name 属性：确定需要设定 JavaBean 的实例名。
- property 属性：确定需要设置的属性名。
- value 属性：确定需要设置的属性值。

getProperty 指令的语法格式如下。

```
<jsp:setProperty name="BeanName" proterty="propertyName" />
```

参数介绍如下。

- name 属性：确定需要输出的 JavaBean 的实例名。
- property 属性：确定需要输出的属性名。

下面通过几段代码来讲解上述动作指令的使用流程。其代码保存在（光盘：源代码/第 10 章/10-6/）中。

新建一个文件 index.jsp，其页面代码如下。

```
<%@ page contentType="text/html; charset=gb2312" language="java" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>javabean 测试</TITLE>
</HEAD>
<BODY>
<!-- 创建 lee.Person 的实例，该实例的实例名为 p1 -->
<jsp:useBean id="p1" class="Person" scope="page"/>
<!-- 设置 p1 的 name 属性值 -->
<jsp:setProperty name="p1" property="name" value="wawa"/>
<!-- 设置 p1 的 age 属性值 -->
```

```
<jsp:setProperty name="p1" property="age" value="23"/>
<!-- 输出 p1 的 name 属性值 -->
<jsp:getProperty name="p1" property="name"/><br>
<!-- 输出 p1 的 age 属性值 -->
<jsp:getProperty name="p1" property="age"/>
${pageScope.p1.age}
</BODY>
</HTML>
```

新建一个文件 Person.java，其代码如下。

```
public class Person
{
   private String name;
   private int age;
   //name 属性对应的 setter 方法
   public void setName(String name)
   {
        this.name = name;
   }
   //age 属性对应的 setter 方法
   public void setAge(int age)
   {
        this.age = age;
   }
   //name 属性对应的 getter 方法
   public String getName()
   {
        return name;
   }
   //age 属性对应的 getter 方法
   public int getAge()
   {
        return age;
   }
}
```

上面的代码十分简单。编码完毕，在浏览器中运行后会得到如图 10-7 所示的效果。

提示：本节知识涉及了 JavaBean 的知识，在第 12 章将会讲解 JavaBean，这里读者不必花太多心思去琢磨这个程序。

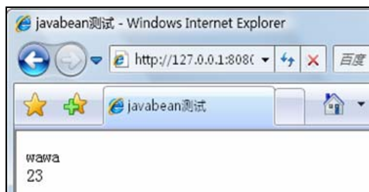


图 10-7　执行后的效果

10.3 JSP 轻松定义自己的标签

自定义标签是一种十分优秀的表现层技术，通过自定义标签库，可以在简单的标签中封装复杂的功能。自定义标签主要用于解决 JSP 和 HTML 的如下 3 个缺点。

- JSP 脚本非常难读。
- JSP 和 HTML 混合造成了维护成本高。
- JSP 嵌套在 HTML 中，美工难以参与开发。

在本节的内容中，将讲解在 JSP 中自定义标签的基本知识。

10.3.1 开发自定义标签类

当在 JSP 页面中使用一个标签时，需要底层处理类来提供支持，这样可以用简单的标签来封装复杂的功能。所有的自定标签类都必须继承一个父类 SimpleTagSupport，继承后需要为标签类编写对应的 getter 和 setter 属性，然后重写 doTag()方法，这个方法用来生成页面的内容。

下面通过一段代码来演示开发自定义标签类的流程，其代码（光盘：源代码/第 10 章/10-7）如下。

```
import java.io.IOException;
package lee;

import javax.servlet.jsp.tagext.*;
import javax.servlet.jsp.*;
import java.io.*;

public class HelloWorldTag extends SimpleTagSupport
{
   //重写 doTag 方法，该方法在标签结束生成页面内容
   public void doTag()throws JspException,
         IOException
   {
         //获取页面输出流，并输出字符串
         getJspContext().getOut().write("大家好，我自定义标签的内容");
   }
}
```

10.3.2 编写 TLD 文件

TLD 是 Tag Library Definition 的缩写，即标签自定义，文件的后缀是 tld。每一个 TLD 都有对应的标签库，一个标签中含有各种各样的标签，TLD 文件也称为标签库定义文件，通过下面的代码可以新建一个 helloworld.tld 文件。

```
<?xml version="1.0" encoding="GBK"?>

<taglib xmlns="http://java.sun.com/xml/ns/j2ee"
```

```
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee web-
jsptaglibrary_2_0.xsd"
    version="2.0">
    <tlib-version>1.0</tlib-version>
    <short-name>mytaglib</short-name>
    <!-- 定义该标签库的 URI -->
    <uri>http://www.leegang.org/mytaglib</uri>

    <!-- 定义第一个标签 -->
    <tag>
        <!-- 定义标签名 -->
        <name>helloWorld</name>
        <!-- 定义标签处理类 -->
        <tag-class>lee.HelloWorldTag</tag-class>
        <!-- 定义标签体为空 -->
        <body-content>empty</body-content>
    </tag>

    <!-- 定义第二个标签 -->
    <tag>
        <!-- 定义标签名 -->
        <name>query</name>
        <!-- 定义标签处理类 -->
        <tag-class>lee.QueryTag</tag-class>
        <!-- 定义标签体为空 -->
        <body-content>empty</body-content>
        <!-- 配置标签属性:driver -->
        <attribute>
            <name>driver</name>
            <required>true</required>
            <fragment>true</fragment>
        </attribute>
        <!-- 配置标签属性:url -->
        <attribute>
            <name>url</name>
            <required>true</required>
            <fragment>true</fragment>
        </attribute>
        <!-- 配置标签属性:user -->
        <attribute>
            <name>user</name>
            <required>true</required>
            <fragment>true</fragment>
        </attribute>
        <!-- 配置标签属性:pass -->
```

```
        <attribute>
            <name>pass</name>
            <required>true</required>

            <fragment>true</fragment>
        </attribute>
        <!-- 配置标签属性:sql -->
        <attribute>
            <name>sql</name>
            <required>true</required>
            <fragment>true</fragment>
        </attribute>
    </tag>

    <!-- 定义第三个标签 -->
    <tag>
        <!-- 定义标签名 -->
        <name>iterator</name>
        <!-- 定义标签处理类 -->
        <tag-class>lee.IteratorTag</tag-class>
        <!-- 定义标签体支持 JSP 脚本 -->
        <body-content>scriptless</body-content>
        <!-- 配置标签属性:collection -->
        <attribute>
            <name>collection</name>
            <required>true</required>
            <fragment>true</fragment>
        </attribute>
        <!-- 配置标签属性:item -->
        <attribute>
            <name>item</name>
            <required>true</required>
            <fragment>true</fragment>
        </attribute>
    </tag>
</taglib>
```

10.3.3 使用标签库里的标签

当布置好环境后，接下来可以编写一段符合需求的 JSP 文件。下面将编写一个简单的 JSP 文件，其代码如下。

```
<%@ page contentType="text/html; charset=GBK"%>
<!-- 导入标签库，指定 mytag 前缀的标签，
   由 http://www.leegang.org/mytaglib 的标签库处理 -->
<%@ taglib uri="http://www.leegang.org/mytaglib" prefix="mytag"%>
<html>
<head>
<title>自定义标签示范</title>
</head>
```

```
<body bgcolor="#ffffc0">
<h2>下面显示的是自定义标签中的内容</h2>
<!-- 使用标签 ，其中 mytag 是标签前缀，根据 taglib 的编译指令，
   mytag 前缀将由 http://www.leegang.org/mytaglib 的标签库处理 -->
<mytag:helloWorld/><BR>
</body>
</html>
```

10.3.4　修改 web.xml

运行自定义标签，确定参数文档后接下来开始修改 web.xml 中的参数，修改后的代码如下。

```
<?xml version="1.0" encoding="GBK"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
   http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
   version="2.5">
</web-app>
```

读者可以从光盘中复制 10–7 文件夹，找到这 4 节的文件源代码以及文件的部署，然后放在 tomcat 服务器的虚拟目录下并运行，将会得到如图 10–8 所示的效果。

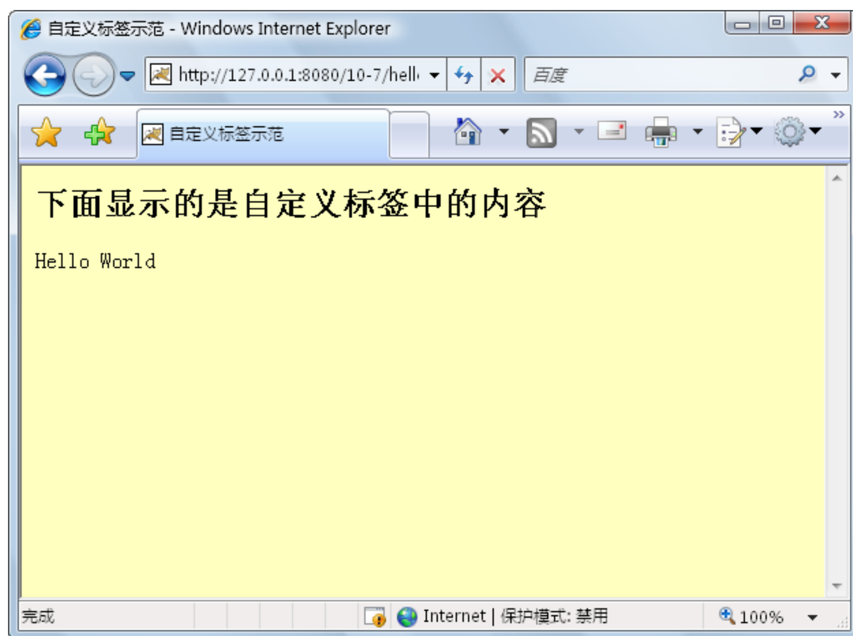


图 10–8　自定义标签

提示： 如果将上述代码放在虚拟目录中，还需要额外配置文件。读者可以将它放置在 Tomcat 6.0\webapps 文件夹下，这个文件一样可以运行正常。

10.3.5 自定义标签的应用

前面的例子中只是讲解了如何创建一个自定义标签的过程，并没有讲解如何给自定义标签定义属性。接下来通过几段代码讲解给自定义标签定义属性的流程，让读者更好地掌握自定义标签，其代码保存在（光盘：源代码/第 10 章/10-8/）中。

新建一个 Java 文件，其代码如下。

```
package lee;
import javax.servlet.jsp.tagext.*;
import javax.servlet.jsp.*;
import java.io.*;
import java.sql.*;
import java.util.*;
public class IteratorTag extends SimpleTagSupport
{
   //标签属性，用于指定需要被迭代的集合
   private String collection;
   //标签属性，指定迭代集合元素，为集合元素指定的名称
   private String item;
   //collection 属性的 setter 和 getter 方法
   public void setCollection(String collection)
   {
        this.collection = collection;
   }
   public String getCollection()
   {
        return this.collection;
   }
   //item 属性的 setter 和 getter 方法
   public void setItem(String item)
   {
        this.item = item;
   }
   public String getItem()
   {
        return this.item;
   }
   //标签的处理方法，简单标签处理类只需要重写 doTag 方法
   public void doTag() throws JspException, IOException
   {
        //从 page scope 中获取属性名为 collection 的集合
        Collection itemList = (Collection)getJspContext().
             getAttribute(collection);
        //遍历集合
        for (Object s : itemList)
        {
             //将集合的元素设置到 page 范围
```

```
                getJspContext().setAttribute(item, s );
                //输出标签体
                getJspBody().invoke(null);
            }
        }
    }
```

新建一个 TLD 文件，其代码如下。

```
<?xml version="1.0" encoding="GBK"?>
<taglib xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee web-
jsptaglibrary_2_0.xsd"
  version="2.0">
  <tlib-version>1.0</tlib-version>
  <short-name>mytaglib</short-name>
  <!-- 定义该标签库的 URI -->
  <uri>http://www.leegang.org/mytaglib</uri>
  <!-- 定义第一个标签 -->
  <tag>
      <!-- 定义标签名 -->
      <name>helloWorld</name>
      <!-- 定义标签处理类 -->
      <tag-class>lee.HelloWorldTag</tag-class>
      <!-- 定义标签体为空 -->
      <body-content>empty</body-content>
  </tag>
  <!-- 定义第二个标签 -->
  <tag>
      <!-- 定义标签名 -->
      <name>query</name>
      <!-- 定义标签处理类 -->
      <tag-class>lee.QueryTag</tag-class>
      <!-- 定义标签体为空 -->
      <body-content>empty</body-content>
      <!-- 配置标签属性:driver -->
      <attribute>
          <name>driver</name>
          <required>true</required>
          <fragment>true</fragment>
      </attribute>
      <!-- 配置标签属性:url -->
      <attribute>
          <name>url</name>
          <required>true</required>
          <fragment>true</fragment>
      </attribute>
      <!-- 配置标签属性:user -->
```

```
        <attribute>
            <name>user</name>
            <required>true</required>
            <fragment>true</fragment>
        </attribute>
        <!-- 配置标签属性:pass -->
        <attribute>
            <name>pass</name>
            <required>true</required>
            <fragment>true</fragment>
        </attribute>
        <!-- 配置标签属性:sql -->
        <attribute>
            <name>sql</name>
            <required>true</required>
            <fragment>true</fragment>
        </attribute>
    </tag>
    <!-- 定义第三个标签 -->
    <tag>
        <!-- 定义标签名 -->
        <name>iterator</name>
        <!-- 定义标签处理类 -->
        <tag-class>lee.IteratorTag</tag-class>
        <!-- 定义标签体支持 JSP 脚本 -->
        <body-content>scriptless</body-content>
        <!-- 配置标签属性:collection -->
        <attribute>
            <name>collection</name>
            <required>true</required>
            <fragment>true</fragment>
        </attribute>
        <!-- 配置标签属性:item -->
        <attribute>
            <name>item</name>
            <required>true</required>
            <fragment>true</fragment>
        </attribute>
    </tag>
</taglib>
```

新建一个 JSP 文件，其代码如下。

```
<%@ page import="java.util.*"%>
<%@ page contentType="text/html; charset=GBK"%>
<!-- 导入标签库，指定 mytag 前缀的标签，
   由 http://www.leegang.org/mytaglib 的标签库处理 -->
<%@ taglib uri="http://www.leegang.org/mytaglib" prefix="mytag"%>
<html>
```

```
    <head>
        <title>带标签体的标签-迭代器标签</title>
    </head>
    <body>
        <h2>带标签体的标签-迭代器标签</h2>
        <hr>
        <%
        //创建一个 List 对象
        List<String> a = new ArrayList<String>();
        a.add("中国");
        a.add("新家坡");
        a.add("日本");
            a.add("韩国");
            a.add("印度");
            a.add("俄罗斯");
        //将 List 对象放入 page 范围内
        pageContext.setAttribute("a" , a);
        %>
        <table border="1" bgcolor="aaaadd" width="300">
        <!-- 使用迭代器标签，对 a 集合进行迭代 -->
        <mytag:iterator collection="a" item="item">
            <tr>
                <td>${pageScope.item}</td>
            <tr>
        </mytag:iterator>
        </table>
    </body>
</html>
```

读者可以从光盘中复制 10-8 文件夹，找到这 4 节的文件源代码以及文件的部署，然后放在 tomcat 服务器的“webapp”目录下，运行后会得到如图 10-9 所示的效果。

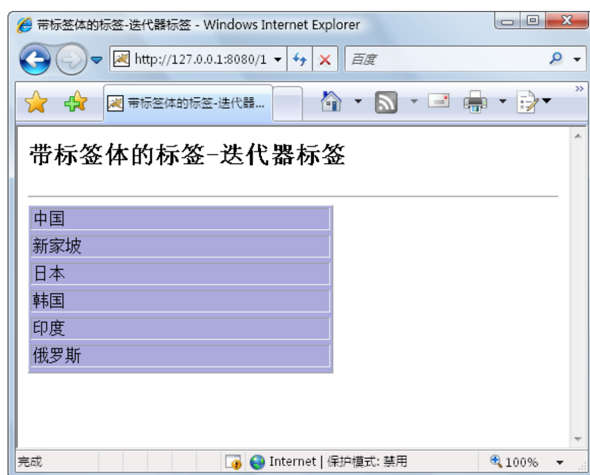


图 10-9　自定义标签的应用

10.4 JSP 2.0 的新特性

目前，大多数 JSP 都是使用 JSP 2.0，很少有人使用 1.0。JSP 功能强大，我们甚至无须学习 Java 即可制作出 Web 页面，这样大大提高了开发效率。后来 Sun 公司又推出的 JSP 2.1 和 JSP 2.0 大同小异，没有太大的区别。

10.4.1 配置 JSP 属性

JSP 属性主要包括如下 4 个方面。

- 是否允许使用表达式语言：可以使用<el-ignored/>元素确定，默认值为 false，即允许使用表达式语言。
- 是否允许使用 JSP 脚本：可以使用<scripting-invalid/>元素确定，默认值为 false，即允许使用 JSP 脚本。
- 是否声明 JSP 页面的编码：可以使用<page-encoding/>元素确定，配置该元素后，可以代替每个页面里 page 指令 contentType 属性的 charset 部分。
- 是否使用隐式包含：可以使用<include-prelude/>和<include-coda/>元素确定，可以代替在每个页面里使用 include 编译指令来包含其他页面。

下面通过一段代码来讲解配置 JSP 属性的流程，其代码保存在（光盘：源代码/第 10 章/10-9/）中，下面开始讲解其具体实现流程。

1）在 Tomcat 服务器下的“webapp”文件夹下新建一个文件夹。

2）在此文件夹中新建一个 WEB-INF 文件夹，然后新建一个 web.xml 文件，其代码如下。

```
<?xml version="1.0" encoding="GBK"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
   http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
   version="2.5">
   <!-- 关于 JSP 的配置信息 -->
   <jsp-config>
         <jsp-property-group>
               <!-- 对哪些文件应用配置 -->
               <url-pattern>/noscript/*</url-pattern>
               <!-- 忽略表达式语言 -->
               <el-ignored>true</el-ignored>
               <!-- 页面编码的字符集 -->
               <page-encoding>GBK</page-encoding>
               <!-- 不允许使用 Java 脚本 -->
               <scripting-invalid>true</scripting-invalid>
               <!-- 隐式导入页面头  -->
               <include-prelude>/inc/top.jspf</include-prelude>
               <!-- 隐式导入页面尾 -->
```

```
                <include-coda>/inc/bottom.jspf</include-coda>
            </jsp-property-group>
            <jsp-property-group>
                <!-- 对哪些文件应用配置 -->
                <url-pattern>*.jsp</url-pattern>
                <el-ignored>false</el-ignored>
                <!-- 页面编码字符集 -->
                <page-encoding>GBK</page-encoding>
                <!-- 允许使用 Java 脚本 -->
                <scripting-invalid>false</scripting-invalid>
            </jsp-property-group>
            <jsp-property-group>
                <!-- 对哪些文件应用配置 -->
                <url-pattern>/inc/*</url-pattern>
                <el-ignored>false</el-ignored>
                <!-- 页面编码字符集 -->
                <page-encoding>GBK</page-encoding>
                <!-- 不允许使用 Java 脚本 -->
                <scripting-invalid>true</scripting-invalid>
            </jsp-property-group>
        </jsp-config>
        <context-param>
            <param-name>author</param-name>
            <param-value>yeeku</param-value>
        </context-param>
    </web-app>
```

3）新建一个 one.jsp 页面，然后输入以下代码。

```
<html>
  <head>
    <title>页面 1</title>
  </head>
  <body>
    <h2>页面 1</h2>
   下面是表达式语言输出：<br>
   ${2008 + 2009}
  </body>
</html>
```

在浏览器中输入网址后将会显示如图 10-10 所示的效果。

4）新建一个 two.jsp 页面，然后输入以下代码。

```
<html>
  <head>
    <title>页面 2</title>
  </head>
```

```
    <body>
      <h1>页面 2</h1>
       下面是表达式语言输出：<br>
     ${1997 + 2342}<br>
       下面是 Scriptlet 输出：<br>
     <%out.println("hello Java");%>
    </body>
  </html>
```

在浏览器中输入网址浏览后将会显示如图 10-11 所示的效果。

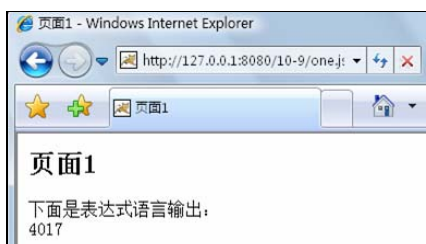


图 10-10　配置后的页面



图 10-11　配置 JSP 属性

10.4.2　JSP 的表达式

表达式语言是一种简单的数据访问方式，使用表达式可以更加方便地访问 JavaBean 和隐含的对象。在编写 JSP 程序的过程中，建议初学者将表达式的格式进行统一规范，并尽量避免使用 Java 脚本。

在 JSP 页面中，用户可以使用 JSP 表达式来实现计算功能，而不必调用 Java 表达式。下面通过一段代码来讲解 JSP 表达式支持算术运算符或逻辑运算符的用法，其代码（光盘：源代码/第 10 章/10-10/）如下。

```
<%@ page contentType="text/html; charset=gb2312"%>
<html>
  <head>
        <title>表达式语言 - 算术运算符</title>
  </head>
  <body>

        <h2>表达式语言 - 算术运算符</h2>
        <hr>

        <table border="1" bgcolor="aaaadd">
             <tr>
                  <td><b>表达式语言</b></td>
```

```
        <td><b>计算结果</b></td>
    </tr>
    <!-- 直接输出常量 -->
    <tr>
        <td>\${1}</td>
        <td>${1}</td>
    </tr>
    <!-- 计算加法 -->
    <tr>
        <td>\${1.2 + 2.3}</td>
        <td>${1.2 + 2.3}</td>
    </tr>
    <!-- 计算加法 -->
    <tr>
        <td>\${1.2E4 + 1.4}</td>
        <td>${1.2E4 + 1.4}</td>
    </tr>
    <!-- 计算减法 -->
    <tr>
        <td>\${-4 - 2}</td>
        <td>${-4 - 2}</td>
    </tr>
    <!-- 计算乘法 -->
    <tr>
        <td>\${21 * 2}</td>
        <td>${21 * 2}</td>
    </tr>
    <!-- 计算除法 -->
    <tr>
        <td>\${3/4}</td>
        <td>${3/4}</td>
    </tr>
    <!-- 计算除法 -->
    <tr>
        <td>\${3 div 4}</td>
        <td>${3 div 4}</td>
    </tr>
    <!-- 计算除法 -->
    <tr>
        <td>\${3/0}</td>
        <td>${3/0}</td>
    </tr>
    <!-- 计算求余 -->
    <tr>
        <td>\${10%4}</td>
        <td>${10%4}</td>
```

```
                </tr>
                <!-- 计算求余 -->
                <tr>
                     <td>\${10 mod 4}</td>
                     <td>${10 mod 4}</td>
                </tr>
                <!-- 计算三目运算符 -->
                <tr>
                     <td>\${(1==2) ? 3 : 4}</td>
                     <td>${(1==2) ? 3 : 4}</td>
                </tr>
           </table>

    <h2>表达式语言 - 逻辑运算符</h2>
       <hr>

      数字之间的比较：
       <table border="1" bgcolor="aaaadd">
       <tr>
         <td><b>表达式语言</b></td>
         <td><b>计算结果</b></td>
      </tr>

      <!-- 直接比较两个数字 -->
       <tr>
         <td>\${1 &lt; 2}</td>
         <td>${1 < 2}</td>
       </tr>
       <!-- 使用 lt 比较运算符 -->
       <tr>
         <td>\${1 lt 2}</td>
         <td>${1 lt 2}</td>
       </tr>
       <!-- 使用>比较运算符 -->
       <tr>
         <td>\${1 &gt; (4/2)}</td>
         <td>${1 > (4/2)}</td>
       </tr>
       <!-- 使用 gt 比较运算符 -->
       <tr>
         <td>\${1 gt (4/2)}</td>
         <td>${1 gt (4/2)}</td>
       </tr>

       <!-- 使用>=比较运算符 -->
       <tr>
```

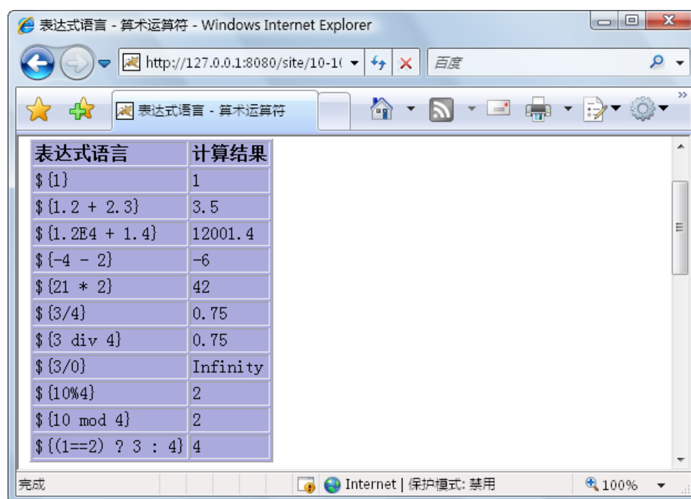
```
    <td>\${4.0 &gt;= 3}</td>
    <td>${4.0 >= 3}</td>
  </tr>
  <!-- 使用 ge 比较运算符 -->
  <tr>
    <td>\${4.0 ge 3}</td>
    <td>${4.0 ge 3}</td>
  </tr>
  <!-- 使用<=比较运算符 -->
  <tr>

    <td>\${4 &lt;= 3}</td>
    <td>${4 <= 3}</td>
  </tr>
  <!-- 使用 le 比较运算符 -->
  <tr>
    <td>\${4 le 3}</td>
    <td>${4 le 3}</td>
  </tr>
  <!-- 使用==比较运算符 -->
  <tr>
    <td>\${100.0 == 100}</td>
    <td>${100.0 == 100}</td>
  </tr>
  <!-- 使用 eq 比较运算符 -->
  <tr>
    <td>\${100.0 eq 100}</td>
    <td>${100.0 eq 100}</td>
  </tr>
  <!-- 使用!=比较运算符 -->
  <tr>
    <td>\${(10*10)!= 100}</td>
    <td>${(10*10)!= 100}</td>
  </tr>
  <!--  先执行运算，再进行比较运算，使用 ne 比较运算符-->
  <tr>
    <td>\${(10*10) ne 100}</td>
    <td>${(10*10) ne 100}</td>
  </tr>

</table>
字符之间的比较：
 <table border="1" bgcolor="aaaadd">
  <tr>
    <td><b>表达式语言</b></td>
    <td><b>计算结果</b></td>
```

```
            </tr>
            <tr>
              <td>\${'a' &lt; 'b'}</td>
              <td>${'a' < 'b'}</td>
            </tr>
            <tr>
              <td>\${'hip' &gt; 'hit'}</td>
              <td>${'hip' > 'hit'}</td>
            </tr>
            <tr>
              <td>\${'4' &gt; 3}</td>
              <td>${'4' > 3}</td>
            </tr>
          </table>
          </body>
       </html>
```

运行后的效果如图 10-12 所示。



表达式语言	计算结果
${1}	1
${1.2 + 2.3}	3.5
${1.2E4 + 1.4}	12001.4
${-4 - 2}	-6
${21 * 2}	42
${3/4}	0.75
${3 div 4}	0.75
${3/0}	Infinity
${10%4}	2
${10 mod 4}	2
${(1==2) ? 3 : 4}	4

图 10-12　表达式

10.5　疑难问题解析

本章，详细讲解了 Java 中标签和动作指令的基本知识。本节中，将对本章中比较难理解的问题进行讲解。

读者疑问：在 JSP 自定义标签里，为什么我建立了 3 个文件，却执行不正常，这是怎么一回事？

解答：首先你要进行 Java 文件编译，生成.class 文件，然后按照一定的规则部署文件，

如 web.xml，如果将所有的文件放在一起，是没有办法执行的。Java Web 开发部署很重要，从小型项目到大型项目，都需要修改或者编辑 web.xml 文档，一定要注意。

读者疑问：在编写的过程中，需要配置 JSP 属性吗？

解答：在一般的开发过程中，不需要配置 JSP 属性，将环境配置好后就可以开发 Java Web 程序了。

读者疑问：Java 的表达式和 JSP 的表达式有什么不同？

解答：它们的原理是相同的，而且结构也是相同的，唯一不同的是，JSP 表达式存在于网页中的 JSP 标签里，而 Java 表达式存在于 java 文件中的方法里。



职场点拨——同事间的距离

进入职场以后，每天要和同事相处。有时和同事相处的时间多于和家人相处的时间，此时正确把握和同事之间的距离就变得愈发重要了。同事间的距离，是一种人际学问。如果和同事刻意保持距离，会让同事认为太冷漠；如果太接近，则可能承担“团伙”的“罪名”。下面笔者给出两条经验之谈，希望能帮助读者正确把握和同事之间的距离。

（1）异性之间——拒绝亲密

工作当中，两性的工作交流非常频繁，并且难免出现异性间的长期合作。虽然“男女授受不亲”的老观念已经废除，但是同事之间也不能过分和异性相处。虽然两性之间的拍档通常比单一性别的合作要来得愉快和谐，但是此时一定要注意，如果你没有意思将这种关系发展为恋情，就应当将感情投入限制在友谊的范围内，即使很有好感，也不应表露出来。如果对方射来丘比特之箭，也应明智地将其化解。千万不要给对方以默许和鼓励。

笔者在此建议：对异性采取大方、不轻浮的态度是同异性工作交往中一个很重要的原则。

（2）同性之间——相依相助

只要你身处办公室，就需要对每个人友好，对同性则更应如此。大家来公司上班均是为了生存，大家同在一个屋檐下，为了一个共同的目标，感受同一种压力，工作中谁也少不了谁，如果可以以一颗同情心来看待同伴的话，关系将很容易处理。又因为是同性，所以很多观点是一致的。在闲暇时刻可以找一些大家均有兴趣的话题和爱好，共同讨论和参与来增加同事间的感情。如果发现同事打小报告，也不必为此而大惊小怪。

笔者在此建议：对同性采取共同进步、相互交流、互相帮助的原则。

第 11 章　Servlet 详解

Servlet 是一种独立于平台和协议的服务器端 Java 应用程序，可以生成动态的 Web 页面。Servlet 担当了 Web 浏览器或其他 HTTP 客户程序发出请求任务，与 HTTP 服务器上的数据库或应用程序之间的中间层。与传统的从命令行启动的 Java 应用程序不同，Servlet 由 Web 服务器进行加载，该 Web 服务器必须包含支持 Servlet 的 Java 虚拟机。通过本章能学到如下知识。

- Servlet 是什么。
- Servlet 的功能。
- HttpServlet 类。
- Servlet 过滤器。
- Servlet 监听。
- 职场点拨——寻兼职。

2010 年 XX 月 XX 日，阴

正当我为钱不够花而发愁时，听说同学小 A 做了一个兼职，赚了相当于他一个月的工资。听后我很是羡慕，看来我也得寻找一种赚钱的门路了……



一问一答

小菜："我是全职工作人员，也能做兼职吗？"

Wisdom："可以，当然是在不影响本职工作的前提下。你可以利用下班后的时间做兼职，不但能给你带来收入，也能拓展你的人脉关系。"

小菜："都是从哪儿获取兼职信息的？"

Wisdom："在我们 IT 行业，特别是程序员，寻找外快的机会很多。在本章的最后，我将概括介绍和兼职有关的一些基本常识。"

小菜："嗯，言归正传，本章的 Servlet 很重要吗？"

Wisdom："Servlet 其实也是 Java 程序，要运行这个 Java 源程序，首先要对它进行编译，然后对它进行环境配置。本章将详细讲解。"

11.1　认识 Servlet

Servlet 其实也是 Java 程序，要运行这个 Java 源程序，首先要对它进行编译，然后对它

进行环境配置。看下面的一段 Servlet 程序，将其运行后可以让读者体会 Servlet 的功能。其代码存放在（光盘：源代码/第 11 章/11-1/）中。首先新建一个 Welcome.java 源程序，其代码如下。

```
package cn.dt.web;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;

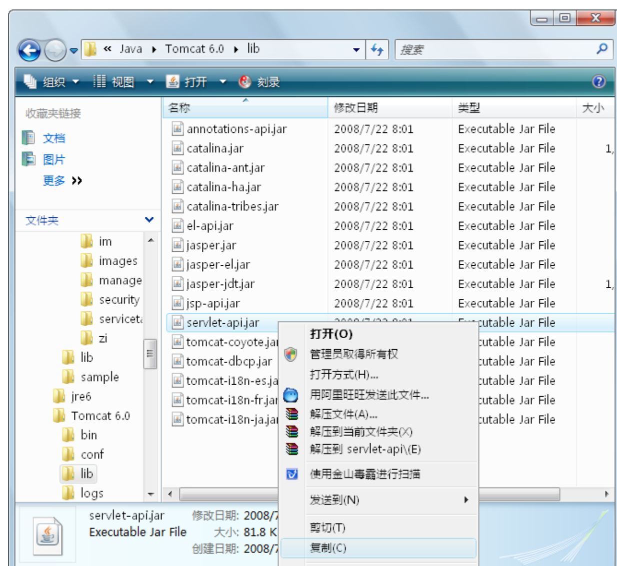
public class Welcome extends HttpServlet
{
   public void doGet(HttpServletRequest request,HttpServletResponse
response)
   throws ServletException,IOException
   {
             response.setContentType("text/html;charset=gb2312");
        PrintWriter out=response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>welcome</title>");
      out.println("</head>");
      out.println("<body>");
        out.println("<h3>");
        out.println("你好！");
        out.println("欢迎你访问该网站！<br>");
        out.println("</body>");
        out.println("<html>");
   }

   public void doPost(HttpServletRequest request,HttpServletResponse
response)
   throws ServletException,IOException
   {
        this.doGet(request,response);
   }
}
```

编写好源代码后并不能对它进行编译，因为 Servlet 不是一般的 Java 程序，除了前面介绍的配置环境外，还需要配置另外的环境。具体配置步骤如下。

1）打开安装 Tomcat 的文件夹，假设安装的目录为“C:\Program Files\Apache Software Foundation\Tomcat 6.0”，然后打开 lib 文件夹，复制“servlet-api.jar”，如图 11-1 所示。

2）打开安装 JDK 的文件夹，假设安装目录为 C:\Program Files\Java\jdk1.6.0_10，然后打开 JDK 文件夹下的“jre\lib\ext”，将复制的“servlet-api.jar”粘贴，如图 11-2 所示。



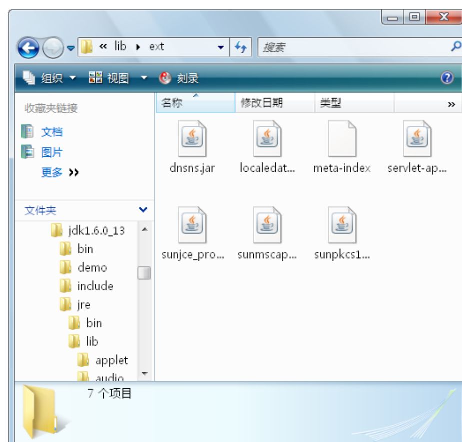


图 11-1　复制 servlet-api.jar 文件　　　　图 11-2　粘贴文件

3）将*.java 进行编译，这里编译文件 TextServlet.java，编译后的效果如图 11-3 所示。

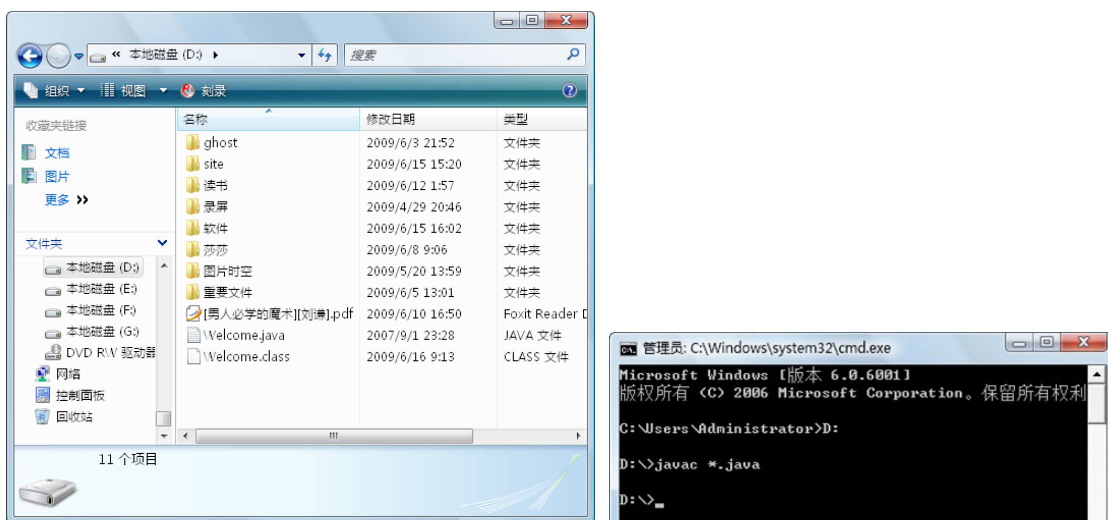


图 11-3　编译文件

4）编译后，在 Tomcat 服务器下的“weapps”文件夹下新建一个名为“11-1”的文件夹，双击打开新建的文件夹，然后建立新的文件夹 WEB-INF，将源文件 Welcome.java 复制到 11-1 的根目录下，如图 11-4 所示。

5）打开 WEB-INF，新建两个文件夹 classes 和 lib，然后新建一个文件 web.xml，如图 11-5 所示。

6）用记事本打开文件 web.xml 并写入代码，然后将其保存，代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
```

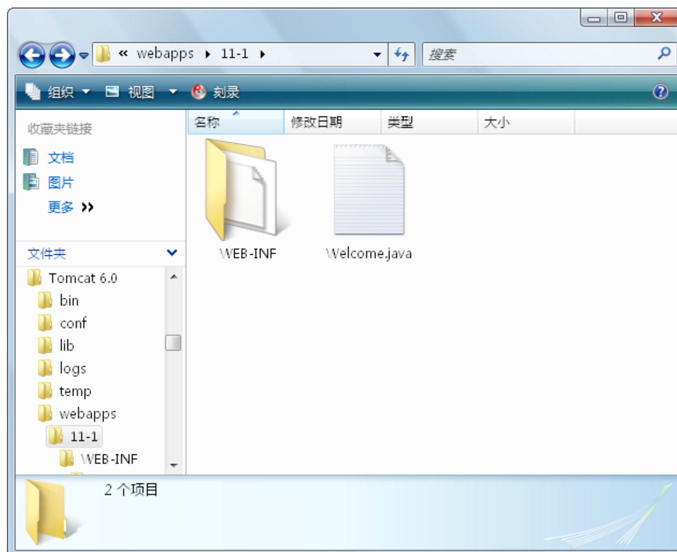


图 11-4　新建文件夹

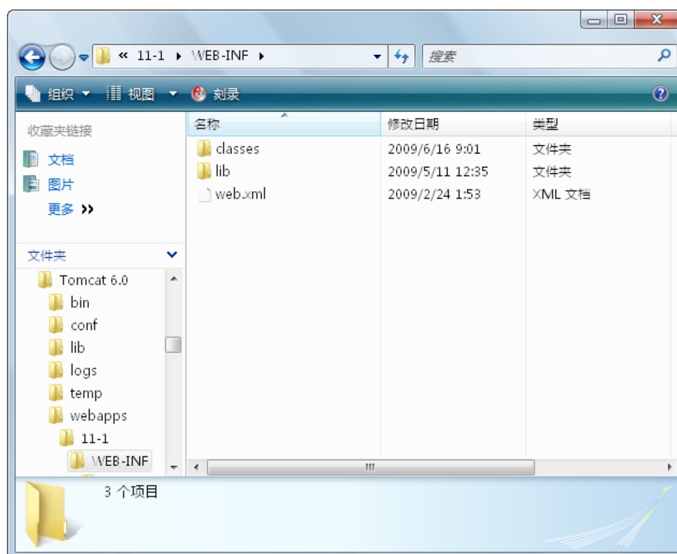


图 11-5　新建配置文件

```
<web-app version="2.4"
   xmlns="http://java.sun.com/xml/ns/j2ee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
   http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <servlet>
    <description>This is the description of my J2EE component
</description>
    <display-name>This is the display name of my J2EE component
```

```
    </display-name>
        <servlet-name>Welcome</servlet-name>
          <servlet-class>cn.dt.web.Welcome</servlet-class>
      </servlet>
      <servlet-mapping>
        <servlet-name>Welcome</servlet-name>
            <url-pattern>/Welcome</url-pattern>
      </servlet-mapping>
      <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
      </welcome-file-list>
    </web-app>
```

7）按照包的路径（package com.wy;）在 classes 中新建几个文件夹，将编译后的扩展名为 class 的文件复制到下面，如图 11-6 所示。

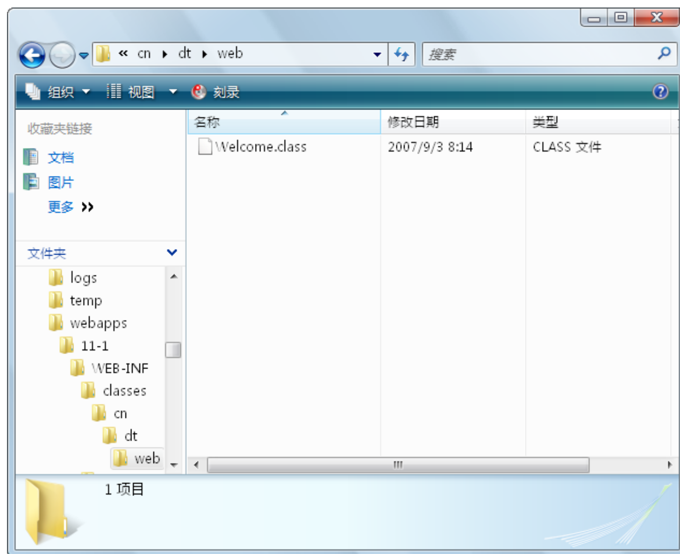


图 11-6　将编译后的文件复制到相应的位置

上述过程就是配置 Servlet 环境的过程，其实整个过程就是编译 Java 源文件并配置环境，然后通过 Java Web 执行源程序内容的过程。执行效果如图 11-7 所示。



图 11-7　执行的结果

11.2　Servlet 是什么

Servlet 是用 Java 编写的 Server 端程序，它与具体的协议和平台无关。Servlet 运行于 Java-enabled Web Server 中。Servlet 可以动态地扩展 Java 的功能，并采用“请求－响应”模式提供 Web 服务。Servlet 和 JSP 相互交互，为开发 Web 服务提供了优秀的解决方案。

11.2.1　Servlet 的技术介绍

Servlet 的功能是，在启用 Java 的 Web 服务器上或应用服务器上运行并扩展了该服务器的功能。Java Servlet 对于 Web 服务器来说，犹如 Java Applet 对于 Web 浏览器。Servlet 装入 Web 服务器并在 Web 服务器内执行，而 Applet 是装入 Web 浏览器并在 Web 浏览器内执行。在 Java Servlet API 中定义了一个 Servlet，这样在 Java 和服务器之间形成了一个标准接口，所以 Servlet 具有跨服务器平台的特性。

当客户机发送请求至服务器时，服务器可以将请求信息发送给 Servlet，并让 Servlet 建立起服务器返回给客户机的响应。当启动 Web 服务器或客户机第一次请求服务时，可以自动装入 Servlet。装入后 Servlet 继续运行直到其他客户机发出请求，Servlet 的功能涉及范围很广，它主要完成下面的工作。

- ❑ 创建并返回一个完整的 HTML 页面，这个页面包含了基于客户请求性质的动态内容。
- ❑ 创建可嵌入到现有 HTML 页面中的一部分 HTML 页面或 HTML 片段。
- ❑ 与其他服务器资源进行通信，包括数据库和基于 Java 的应用程序。
- ❑ 使用多个客户机处理连接，接收多个客户机的输入，并将结果广播到多个客户机上。例如，Servlet 可以是多参与者的游戏服务器。
- ❑ 当允许在单连接方式下传送数据的情况下，在浏览器上打开服务器至 Applet 的新连接，并将该连接保持在打开状态。当在允许客户机和服务器简单、高效地执行会话的情况下，Applet 也可以启动客户浏览器和服务器之间的连接。可以通过定制协议或标准（如 IIOP）进行通信。
- ❑ 对特殊的处理采用 MIME 类型过滤数据。
- ❑ 将定制的处理提供给所有服务器的标准例行程序。例如，Servlet 可以修改如何认证用户。

11.2.2　Servlet 技术的优越性

Servlet 是 Java 技术对 CGI 编程的回答，Servlet 程序在服务器端运行，可以动态地生成 Web 页面。与传统的 CGI 和许多其他类似 CGI 的技术相比，Java Servlet 具有更高的效率，更容易使用，功能更强大，具有更好的可移植性，更节省投资。Servlet 具有以下优点。

（1）高效

在 Servlet 中，每个请求由一个轻量级的 Java 线程处理（而不是重量级的操作系统进程）。在传统 CGI 中，如果有 N 个并发的对同一 CGI 程序的请求，则该 CGI 程序的代码在内存中重复装载了 N 次；而在 Servlet 中，处理请求的是 N 个线程，只需要一份 Servlet 类代码即可实现。在性能优化方面，Servlet 也比 CGI 有更多的选择，比如缓冲以前的计算结

果，保持数据库连接的活动等。

（2）方便

Servlet 提供了大量的实用工具例程，例如，自动解析和解码 HTML 表单数据、读取和设置 HTTP 头、处理 Cookie、跟踪会话状态等，这样大大方便了程序员。

（3）功能强大

在 Servlet 中，可以轻松地完成许多 CGI 程序很难完成的任务。例如，Servlet 能够直接和 Web 服务器交互，而普通的 CGI 程序不能。Servlet 还能够在各个程序之间共享数据，使得数据库连接池之类的功能很容易实现。

（4）可移植性好

Servlet 用 Java 编写，在 Servlet API 中提供了完善的标准。所以编写的 Servlet 程序无需任何实质上的改动，都可以移植到 Apache、Microsoft IIS 或者 WebStar。几乎所有的主流服务器都直接或通过插件支持 Servlet。

11.2.3 Servlet 持久性

很多专家都在考虑 Servlet 在 Java 程序能活多久的问题，究竟它的生命周期是怎么一回事？Servlet 的生命周期始于将它装入 Web 服务器的内存时，在终止或重新装入 Servlet 时结束。Servlet 的持久性主要分为初始化、请求处理、终止这几个阶段。

11.3 开发一个 Servlet 程序

在 11.1 一节中其实已经介绍了配置并发布 Servlet 程序的过程，鉴于 Servlet 的重要性和难度，在本节将进一步讲解 Servlet 程序的编译和发布过程。编写 Servlet 程序十分简单，读者通过下面内容的学习就会编写和部署一个 Servlet 的程序，从而进一步认识 Servlet 程序的功能。

11.3.1 编写一个 Servlet 源程序

要执行 Servlet 源程序，首先必须编写一个 Java 源文件让它实现一些功能。下面将编写一个简单的源代码，在代码中使用了 Servlet 的功能。其代码（光盘：源代码/第 11 章/11-2/）如下。

```
package com.wy;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class TextServlet extends HttpServlet
{
  public void doGet(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException
```

```
        {
                response.setContentType("text/html;charset=gb2312");
                PrintWriter out = response.getWriter();
                out.println("<HTML>");
                out.println("  <HEAD><TITLE>第一个 Servlet</TITLE></HEAD>");
                out.println("  <BODY background=images/b01.jpg>");
                out.println("Servlett 是这样诞生的！");
                out.println("  </BODY>");
                out.println("</HTML>");
                out.flush();
                out.close();
          }
          public void doPost(HttpServletRequest request, HttpServletResponse
        response)
                      throws ServletException, IOException {
                doGet(request, response);
          }
        }
```

11.3.2　编译 Servlet 文件

Servlet 文件不是一般的 Java 文件，JDK 不能对它进行直接编译，必须配置专门的环境才能编译 Servlet 文件。编译 Servlet 文件的具体操作流程如下。

1）打开安装过 Tomcat 的文件夹，假如安装的目录为“C:\Program Files\JavaTomcat 6.0”，然后打开 lib 文件夹，将“servlet-api.jar”复制，如图 11-8 所示。

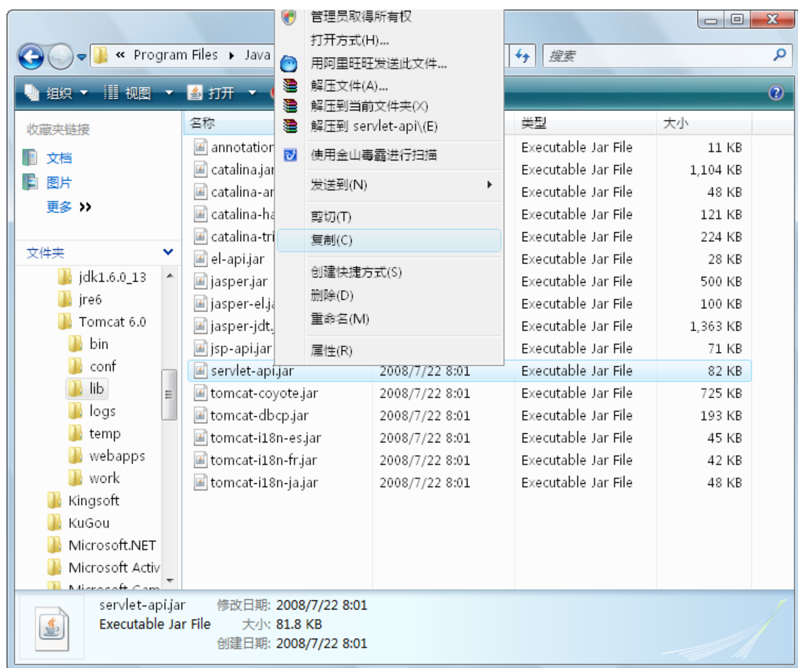


图 11-8　复制文件

2）打开安装 JDK 的文件夹，例如，安装目录为“C:\Program Files\Java\ jdk1.6.0_10”，然后打开 JDK 文件夹下的“jre\lib\ext”，将“servelet-api.jar”粘贴在此，如图 11-9 所示。

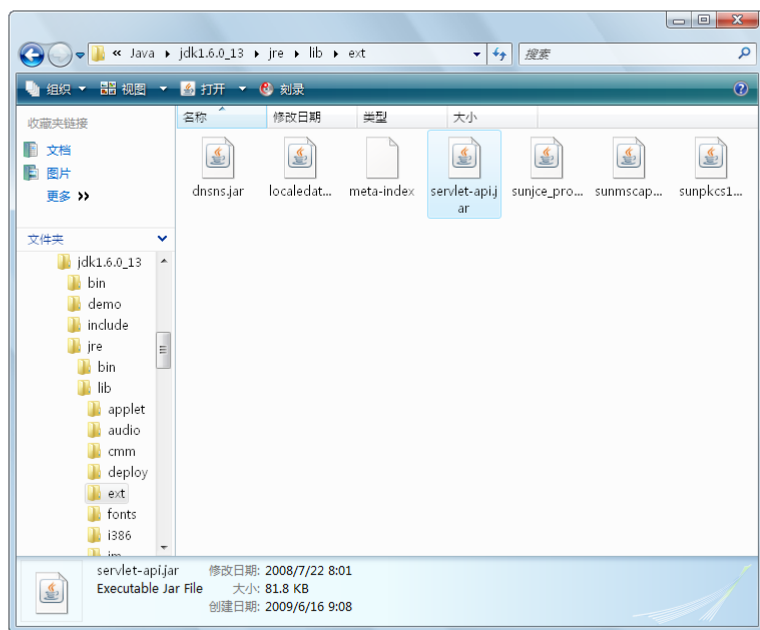


图 11-9　粘贴文件

3）将 TextServlet.java 编译，编译后的效果如图 11-10 所示。

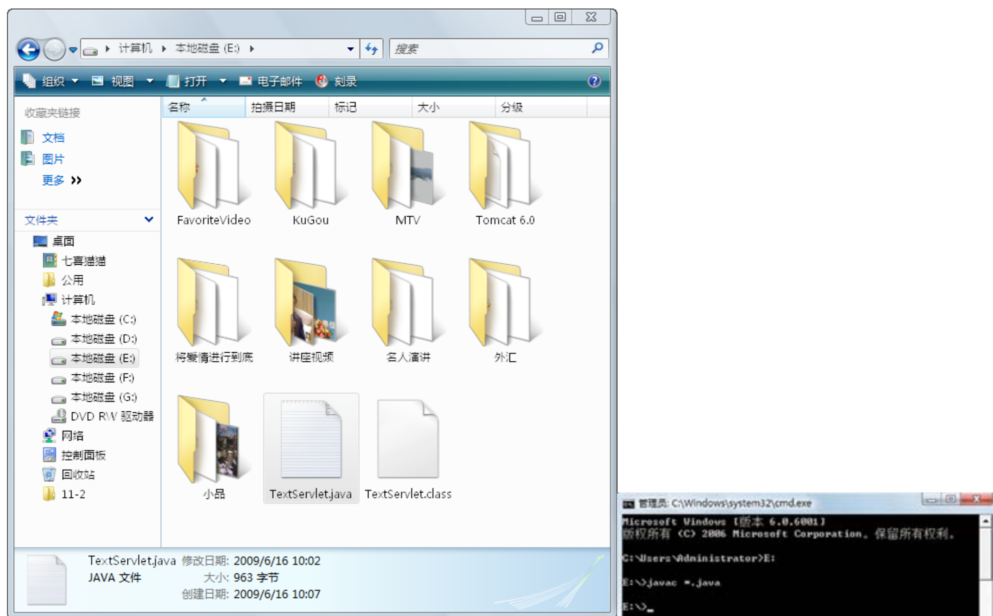


图 11-10　编译源文件

11.3.3　部署 Servlet 运行环境

1）打开 Tomcat 6.0\webapps，假设是“C:\Program Files\java\Tomcat 6.0\webapps”，然后

新建一个文件夹，假如将文件夹命名为“11–2”。在此可以新建几个文件夹，将文件 TextServlet.java 复制到“11–2”文件夹（可以不复制，这和运行没有关系）中，然后新建两个文件夹，如图 11–11 所示。

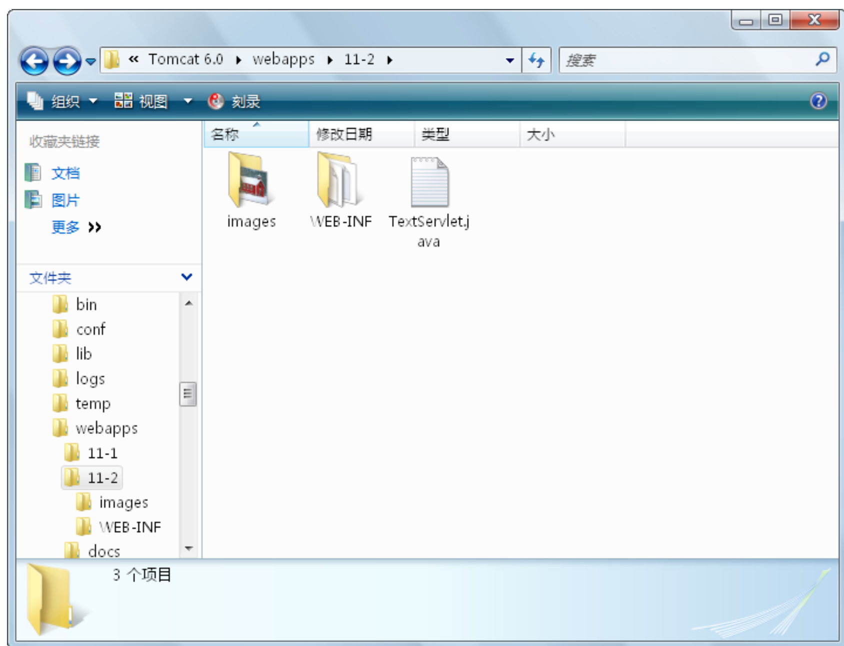


图 11–11　新建文件夹

2）将图片文件“b01.jpg”复制到“images”文件夹（用来显示背景图，与上一节的程序代码“out.println("<BODY background=images/b01.jpg>")”一一对应），然后打开 WEB–INF 目录，新建两个文件夹“classes”和“lib”，然后新建一个文件 web.xml，如图 11–12 所示。

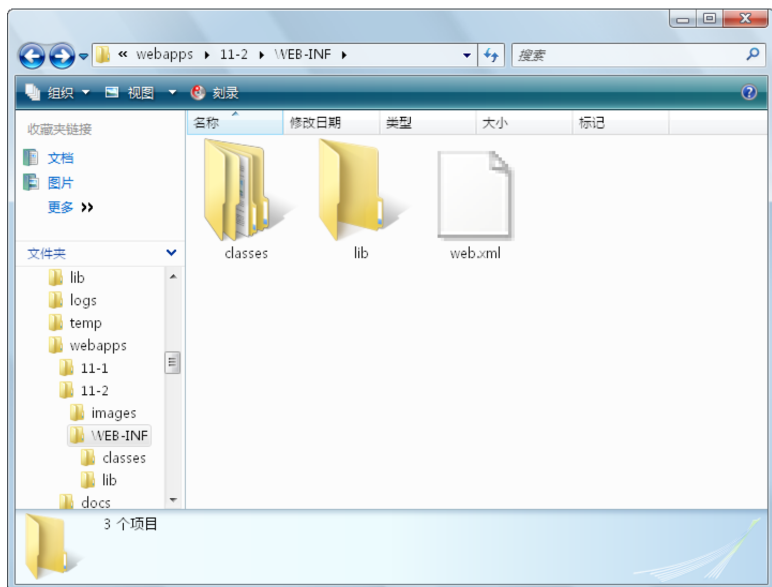


图 11–12　新建文件夹配置文件

3）用记事本打开 web.xml 并写入代码，然后将其保存，代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4"
   xmlns="http://java.sun.com/xml/ns/j2ee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
   http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <servlet>
    <description>This is the description of my J2EE component
</description>
    <display-name>This is the display name of my J2EE component
</display-name>
    <servlet-name>TextServlet</servlet-name>//
    <servlet-class>com.wy.TextServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>TextServlet</servlet-name>
    <url-pattern>/TextServlet</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

4）按照包的路径（package com.wy;）在 classes 新建几个文件夹，将编译后的扩展名为 class 的文件复制到里面，如图 11-13 所示。

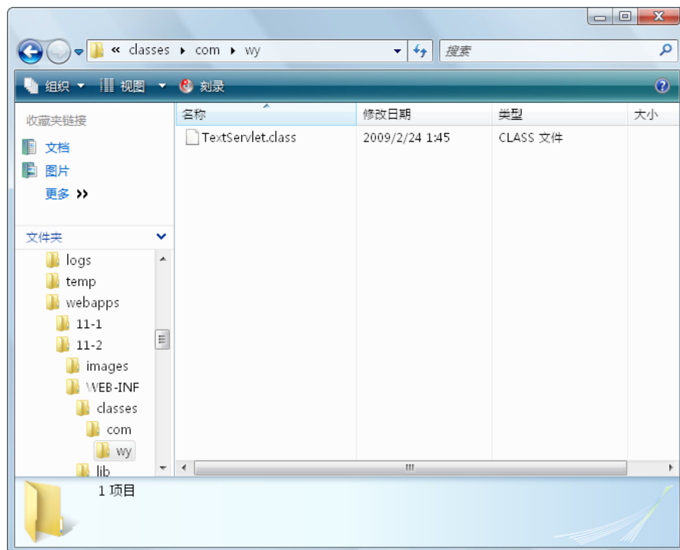


图 11-13 复制对应的文件

5）在浏览器地址栏输入 http://127.0.0.1:8080/11-2/TextServlet，运行效果如图 11-14 所示。

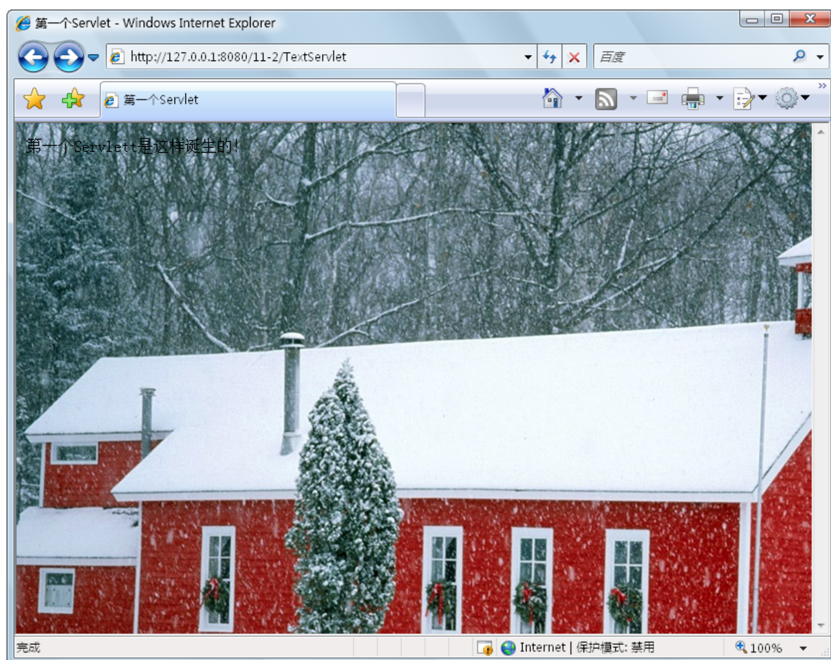


图 11-14　浏览的结果

提示：因为 Servlet 和 JSP 是完全统一的，所以 Servlet 和 JSP 有相同的生命周期。

11.4　Servlet 的功能

Servlet 的功能倒底有多强大，取决于 Java 为它提供了多少接口和类。在本节的内容中，将详细讲解在 Servlet 中使用频率较高的常用接口和类。

11.4.1　与 Servlet 配置相关的接口

和 Servlet 配置有关的接口是 java.servlet.ServletCongfig 接口，下面是一段简单的 Servlet 配置代码。

```
      <servlet-class>com.wy.TextServlet
    </servlet-class>
      </servlet>
      <servlet-mapping>
        <servlet-name>TextServlet</servlet-name>
        <url-pattern>/TextServlet</url-pattern>
      </servlet-mapping>
```

读者可以阅读 Java API 手册，里面列举了 Java 的常用接口和类。在 Servlet 中是通过其中的方法实现具体功能的，常用方法的具体说明如下。

- getinitParameter（String name）：返回指定名称的参数的字符串。
- getinitparameterName()：返回 Servlet 所有初始化的值。

- getServletcontext()：返回当前 Servlet 正在执行的上下文的引用。
- getServletName()：返回当前 Servlet 实例的名称。
- getcontext（String uripath）：返回由参数指定的一个对象，
- log（String msg）：向 Servelt 日志写入信息。
- getServetinfo()：返回当前运行的 Servelt 容器名称和版本代码。
- getattribute（String name）：返回由参数指定的名称。
- getAttributeNames()：返回由容器中所有属性指定的名称。
- getServletContextName：返回当前 Web 应用程序的 ServletContext 的名称。

11.4.2 Servlet 编程接口

在 Java 程序设计中，Servlet 编程需用引用 Javax.servlet 包和接口，下面将详细讲解常用 Servelt 编程接口的基本知识。

1．初始化 Servlet

在 Web 容器加载和实例化 Servlet 类之后，在 Servlet 实例传递来自客户端的请求之前，Web 容器会对 Servlet 进行初始化。用户可以自定义这个初始化过程，以允许 Servlet 配置数据、初始化资源，即不中断数据源。初始化 Servlet 的格式如下。

```
public void init（ServletCongfig arg0）throws servletException
```

2．Servlet 业务实现

在编程的过程中经常需要实现 Servlet 业务，实现 Servlet 业务的格式如下。

```
Public void service（ServletRequst arg0,ServletResponse arg）throws
 servletException,IOException
```

3．Servlet 销毁

当执行完毕不再需要一些内容时，这时候就需要执行销毁这个动作，销毁 Servlet 的格式如下。

```
public void destroy()
```

4．Servlet 和 Web 容器进行通信

Servlet 与 Web 通信的方法十分简单，该方法的格式如下。

```
public servletconfig getServletConfig()
```

5．获取 Servlet 的基本信息

获取 Servlet 基本信息的格式如下。

```
public servlet getServletInfo()
```

11.4.3 Servlet 应用接口

在上一节的内容中，已经讲解了 Servlet 编程接口的基本知识。下面将通过几段代码来讲解如何应用这些接口，其代码保存在（光盘：源代码/第 11 章/11-3）中。程序文件

TextServlet.java 的具体代码如下。

```
package com.wy;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class TextServlet implements Servlet
 {
   public void init(ServletConfig arg0) throws ServletException
{
        System.out.println("执行 init()方法");
   }
   public void service(ServletRequest arg0, ServletResponse arg1)
             throws ServletException, IOException {
        System.out.println("执行 service()方法");
   }
   public void destroy()
 {
        System.out.println("destroy()方法");
   }
   public ServletConfig getServletConfig()
{
        return null;
   }
   public String getServletInfo() {
        return null;
   }
}
```

配置文件 web.xml 的实现代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4"
   xmlns="http://java.sun.com/xml/ns/j2ee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
   http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <servlet>
    <description>This is the description of my J2EE component
</description>
    <display-name>This is the display name of my J2EE component
</display-name>
    <servlet-name>TextServlet</servlet-name>
    <servlet-class>com.wy.TextServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>TextServlet</servlet-name>
```

```
        <url-pattern>/TextServlet</url-pattern>
      </servlet-mapping>
    </web-app>
```

上面这个程序为空，在浏览器输入地址 http://localhost:8080/11-3/TextServlet 后得到的结果也是一个空白页面，如图 11-15 所示。

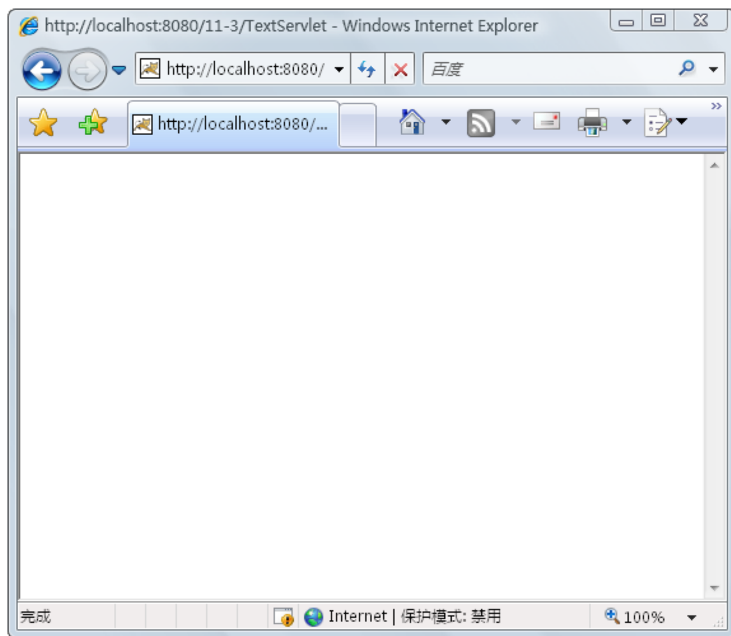


图 11-15　执行结果

如果用控制器查看的话，第一次会看到如下结果。

执行 init()方法

执行 service()方法

如果重新访问网页一次，控制器会出现如下结果。

执行 init()方法

执行 service()方法

执行 service()方法

11.5　使用 HttpServlet 类封装请求

Servlet 的请求可以包含多个数据项，当 Web 容器接收到 Servlet 的请求时，容器就会将请求封装成一个对象。在本节的内容中，将详细讲解使用 HttpServlet 类封装请求的基本知识。

11.5.1　Httpservlet 类

类 HttpSerlet 是针对 HTTP 的 Web 服务器的 Servlet 类，它是对 GenericServlet 类的一个

扩展，同时又是一个抽象类。在一个 HttpServlet 子类中必须至少重载以下方法中的一个。

- doGet()方法，适用于 HTTP GET 请求。
- doPost()方法，适用于 HTTP POST 请求。
- doPut()方法，适用于 HTTP PUT 请求。
- doDelete()方法，适用于 HTTP DELETE 请求。
- init()和 destroy()方法，管理 Servlet 生命周期中的资源。
- getServletInfo()方法，提供 Servlet 本身的信息。

类 HttpServlet 中的 getServletConfig()、getServletContext()、getServletInfo()、getServletName()、log()等方法和 GenericServlet 类中的同名方法功能相同，本节不再赘述。

HttpSerlet 类的构造方法如下。

```
public HttpServlet()
```

接下来将通过几段程序来讲解 HttpSerlet 类的使用过程，其代码（光盘：源代码/第 11 章/11-4/）如下。

```
package com.wy;
import java.io.*;
import java.sql.*;
import javax.servlet.ServletException;
import javax.servlet.http.*;
public class InitServlet extends HttpServlet
 {
   private String driver = "";
   private String URL = "";
   private String username = "";
   private String password = "";
   public void init() throws ServletException
{
         driver = getInitParameter("driver");
         URL = getInitParameter("URL");
         username = getInitParameter("username");
         password = getInitParameter("password");
   }
//获得数据库连接的方法
   public Connection getConnection() {
         Connection con = null;
         try {
              Class.forName(driver);
              con = DriverManager.getConnection(URL, username,
password);
              System.out.print("2");
         } catch (Exception e) {
         }
         return con;
```

```
    }
//通过 doGet()方法实现数据查询功能
    public void doGet(HttpServletRequest request, HttpServletResponse
response)
                throws ServletException, IOException {
        response.setContentType("text/html;charset=gb2312");
        PrintWriter out = response.getWriter();
        Connection con = this.getConnection();
        try {
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery("select * from
tb_user");
            while (rs.next()) {
                out.print(rs.getString("id"));
                out.print("    ");
                out.print(rs.getString("name"));
                out.print("    ");
                out.print(rs.getString("age"));
                out.print("    ");
                out.print(rs.getString("sex"));
                out.print("<br>");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    public void doPost(HttpServletRequest request, HttpServletResponse
response)
                throws ServletException, IOException {
        doGet(request, response);
    }
}
```

新建文件 web.xml，其配置代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
    <servlet>
        <description>
            This is the description of my J2EE component
        </description>
        <display-name>
            This is the display name of my J2EE component
        </display-name>
```

```
            <servlet-name>InitServlet</servlet-name>
            <servlet-class>com.wy.InitServlet</servlet-class>
            <init-param>
                 <param-name>driver</param-name>
                 <param-value>
                      com.microsoft.jdbc.sqlserver.SQLServerDriver
                 </param-value>
            </init-param>
            <init-param>
                 <param-name>URL</param-name>
                 <param-value>
                      jdbc:microsoft:sqlserver://localhost:1433;Database
Name=db_database19
                 </param-value>
            </init-param>
            <init-param>
                 <param-name>username</param-name>
                 <param-value>sa</param-value>
            </init-param>
            <init-param>
                 <param-name>password</param-name>
                 <param-value></param-value>
            </init-param>
   </servlet>
   <servlet-mapping>
            <servlet-name>InitServlet</servlet-name>
            <url-pattern>/InitServlet</url-pattern>
   </servlet-mapping>
</web-app>
```

此时可以按照前面介绍的方式进行编译与配置。但是执行后会出错，因为还没有建立数据库。有关数据库的知识，将在本书后面的章节中进行讲解，上述代码的目的是让读者理解 HttpServlet 的基本知识。

11.5.2　请求与响应

请求与响应一般都是使用表单与服务器发生交互的，可以使用接口 httpServletRequst、httpServletRespons 实现请求与响应，在里面提供了很多方法，读者可以查看 Java API 手册，这里不再赘述。下面通过几段代码来讲解请求与响应的过程，其代码保存在（光盘：源代码/第 11 章/11-5/）中，流程如下。

新建一个 java 文件，其代码如下。

```
package co.cm.ino;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
```

```
import java.io.*;
public class studentInfo extends HttpServlet
{
   public void doGet(HttpServletRequest request,HttpServletResponse
response)
                  throws ServletException,IOException
     {
        request.setCharacterEncoding("gb2312");
        response.setContentType("text/html;charset=gb2312");
        PrintWriter out=response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>基本信息</title>");
        out.println("</head>");
        out.println("<body>");
      out.println("<h2>学生的基本信息</h2>");
       out.println("<h3>姓名:"+request.getParameter("sname")+"</h3>");
        out.println("<h3>学号:"+request.getParameter("snumber")+
"</h3>");
        out.println("<h3>班级:"+request.getParameter("sclass")+
"</h3>") ;
        out.println("</body>");
        out.println("</html>");
     }
    public void doPost(HttpServletRequest request,HttpServletResponse
response)
   throws ServletException,IOException
   {
                   this.doGet(request,response);
   }
}
```

编写 Info.html 文件，其代码如下。

```
<html>
    <head>
         <title></title>
    </head>
    <body>
        <form action="studentInfo" name="myform" method="post">
              姓名:<input type="text" name="sname"><br>
              学号:<input type="text" name="snumber"><br>
              班级:<input type="text" name="sclass"><br>
                    <input type="submit" value="提交">
        </form>
    </body>
</html>
```

编写配置文件 web.xml，其代码如下：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.
sun.com/xml/ns/j2ee/web-app_2_4.xsd"
   version="2.4">
   <servlet>
      <servlet-name>studentInfo</servlet-name>
      <servlet-class>co.cm.ino.studentInfo</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>studentInfo</servlet-name>
      <url-pattern>/studentInfo</url-pattern>
   </servlet-mapping>
  </web-app>
```

上面这个程序代码为空，在浏览器中输入地址 http://localhost:8080/11-5/Info.html 运行后得到如图 11-16 所示结果。

将自己填入的表单信息进行提交，交给 Servlet 处理后会得到如图 11-17 所示的效果。

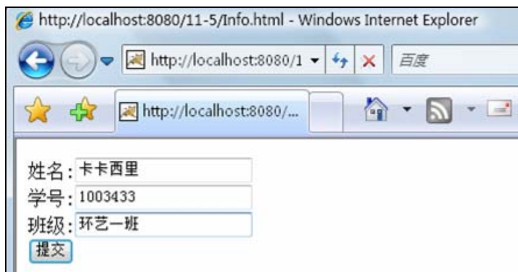


图 11-16　将信息输入表单



图 11-17　输出的信息

11.5.3　会话

会话是客户端和 HTTP 两者之间通过 HttpSession 接口实现的关联，这种关联可以维持指定的时间，同时可以跨越多个连接或用户界面请求。一个 Session 只负责和一个用户通信，并存储该用户的有关信息。在会话中有如下常用的方法。

- public void setAttribute（String name, Object value）：将 value 对象以 name 名称绑定到会话。
- public object getAttribute（String name）：取得 name 的属性值，如果属性不存在则返回 null。
- public void removeAttribute（String name）：从会话中删除 name 属性，如果不存在则不会执行，也不会抛出错误。
- public Enumeration getAttributeNames()：返回和会话有关的枚举值。

- public void invalidate()：使会话失效，同时删除属性对象。
- public Boolean isNew()：用于检测当前客户是否为新的会话
- public long getCreationTime()：返回会话创建时间。
- public long getLastAccessedTime()：返回在会话时间内 Web 容器接收到客户最后发出的请求时间。
- public int getMaxInactiveInterval()：返回在会话期间内客户请求的最长时间，单位是 s。
- public void setMasInactiveInterval（int seconds）：允许客户请求的最长时间
- ServletContext getServletContext()：返回当前会话的上下文环境，ServletContext 对象可以使 Servlet 与 Web 容器进行通信。
- public String getId()：返回会话期间的识别号。

下面通过几段代码来讲解会话的实现流程，其代码（光盘：源代码/第 11 章/11-6/）如下。

```
package cf.re.wyf;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
public class StudentInfo extends HttpServlet
{
  public void doGet(HttpServletRequest request,
               HttpServletResponse response)
               throws ServletException,IOException
     {
        request.setCharacterEncoding("gb2312");
       response.setContentType("text/html;charset=gb2312");
       PrintWriter out=response.getWriter();

       String sname=request.getParameter("sname");
       String snumber=request.getParameter("snumber");
       String sclass =request.getParameter("sclass");

       HttpSession session=request.getSession();
       session.setAttribute("name",sname);
       session.setAttribute("number",snumber);
       session.setAttribute("stclass",sclass);

       out.println("<html>");
       out.println("<head>");
       out.println("<title>会话相关</title>");
       out.println("</head>");
       out.println("<body>");
       out.println("这是一个有关会话相关的页面");
```

```
            out.println("</body>");
            out.println("</html>");
        }
        public void doPost(HttpServletRequest request,
                        HttpServletResponse response)
                    throws ServletException,IOException
        {
            this.doGet(request,response);
        }
}
```

新建文件 ShowInfo.jsp，其代码如下。

```
<%@
    page contentType="text/html;charset=gb2312"
%>
<html>
    <head>
        <title>显示信息</title>
    </head>
    <body>
        <h2>以下是相关的信息</h2>
        <%
         String a=(String)session.getAttribute("name");
         String b=(String)session.getAttribute("number");
         String c=(String)session.getAttribute("stclass");
         out.println("姓名:"+a+"<br>");
         out.println("学号:"+b+"<br>");
         out.println("班级:"+c+"<br>");
         %>
    </body>
</html>
```

配置文件 web.xml 的代码如下。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.
sun.com/xml/ns/j2ee/web-app_2_4.xsd"
   version="2.4">
   <servlet>
      <servlet-name>StudentInfo</servlet-name>
      <servlet-class>cf.re.wyf.StudentInfo</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>StudentInfo</servlet-name>
```

```
            <url-pattern>/StudentInfo</url-pattern>
        </servlet-mapping>
    </web-app>
```

在浏览器中输入网址 http://localhost/20-6/Info.html，可以得到如图 11-18 所示的页面。

单击“提交”按钮，与上一节的代码不同，并没有显示自己提交的信息，却得到了如图 11-19 所示的效果。

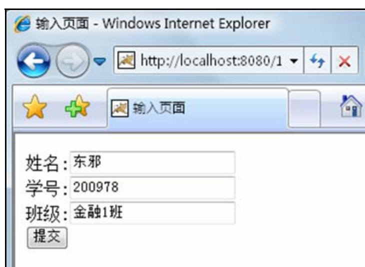


图 11-18　填写表单信息

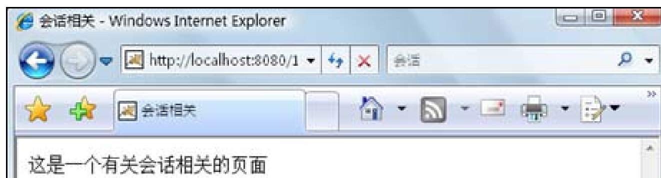


图 11-19　提交后的信息

在浏览器中输入网址 http://localhost:8080/11-6/ShowInfo.jsp，得到如图 11-20 所示效果。

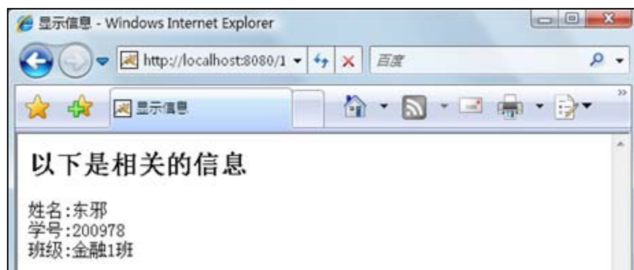


图 11-20　会话结果

11.5.4　轻松将请求进行封装

Servlet 的请求可以包含多个数据项，当 Web 容器接收到 Servlet 的请求时，容器就会将请求封装成一个对象。在 Java 中通常需要使用 doGet()方法和 doPost()方法实现 HttpServlet。

1. doGet()方法

doGet()方法用来响应浏览器使用的 get 方法提取数据的请求，使用 get 方法提取的数据会在浏览器的地址栏中显示正在传给 Servlet 的参数及其值，这样做会给系统安全方面带来一些问题。下面通过一段代码来讲解使用 doGet()方法的流程，其代码（光盘：源代码/第 11 章/11-7/）如下。

```
package dw;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
import java.io.*;
```

```
public class usingdoGet extends HttpServlet
{
   public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
                   throws ServletException,IOException
     {
        request.setCharacterEncoding("gb2312");
        response.setContentType("text/html;charset=gb2312");
        PrintWriter out=response.getWriter();
        String usingname=request.getParameter("usingname");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>用户登录</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<br>");
        out.println("<br>");
      if(usingname!=null && !usingname.equals(""))
      {
        out.println("欢迎你的访问:"+usingname);
      }
      else
      {
        out.println("请重新输入");
      }
        out.println("</body>");
        out.println("</html>");
     }
    public void doPost(HttpServletRequest request,
                   HttpServletResponse response)
                   throws ServletException,IOException
   {

        this.doGet(request,response);
   }
}
```

文件 index.html 的代码如下。

```
<html>
    <head>
         <title>应用 doGet</title>
    </head>
    <body>
         <p>请输入你的昵称:</p>
         <form action="usingdoGet" method="post">
             <input type="text" name="usingname"><br>
```

```
                    <input type="submit" value="确定">
          </form>
     </body>
</html>
```

配置文件 web.xml 的具体代码如下。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.
sun.com/xml/ns/j2ee/web-app_2_4.xsd"
   version="2.4">
    <servlet>
      <servlet-name>usingdoGet</servlet-name>
      <servlet-class>dw.usingdoGet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>usingdoGet</servlet-name>
      <url-pattern>/usingdoGet</url-pattern>
   </servlet-mapping>
    </web-app>
```

将上述 Java 文件进行编译，然后将其复制到包规定的文件夹下，启动 Tomcat，输入地址 http://localhost:8080/11-7/，运行后会得到如图 11-21 所示的效果。

填写信息并单击“确定”按钮提交，得到如图 11-22 所示的效果。



图 11-21　填写表单

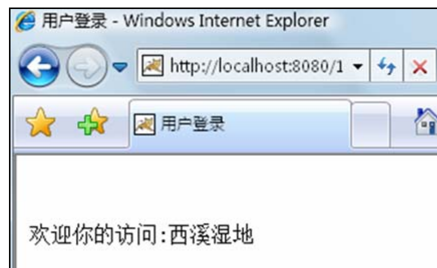


图 11-22　最终效果

2．doPost()方法

doPost()方法能够把数据从客户端传送到服务器端，好处是可以隐藏传送给服务器的任何数据。此方法适合发送大量的数据，下面通过一段代码来讲解使用 doPost()方法的流程，其代码（光盘：源代码/第 11 章/11-8/）如下。

```
package co;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
import java.io.*;
```

```
public class usingdoPost extends HttpServlet
{

   public void doGet(HttpServletRequest request,
                 HttpServletResponse response)
                  throws ServletException,IOException
      {
         request.setCharacterEncoding("gb2312");
         response.setContentType("text/html;charset=gb2312");
         PrintWriter out=response.getWriter();
         out.println("<html>");
         out.println("<head>");
         out.println("<title></title>");
         out.println("</head>");
         out.println("<body>");
               out.println("<br>");
         out.println("<h2>我是通过 Dopost 传递写给它的话的:</h2>");
       out.println("<textarea name=usingtext cols=40 rows=4>");
       out.println(request.getParameter("usingtext"));
       out.println("</textarea>");
         out.println("</body>");
         out.println("</html>");
      }
    public void doPost(HttpServletRequest request,
                  HttpServletResponse response)
                  throws ServletException,IOException
   {

         this.doGet(request,response);
   }
}
```

新建一个 index.html 页面，代码如下。

```
<html>
    <head>
          <title>写给它的话</title>
    </head>
    <body>
          <form action="usingdoPost" method="post">
              <h3>请输入你想要说的话:</h3>
              <textarea name="usingtext"cols="40" rows="4">
</textarea><br>
                    <input  type="submit" value="确定">
          </form>
    </body>
</html>
```

配置文件 web.xml，具体代码如下。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.
sun.com/xml/ns/j2ee/web-app_2_4.xsd"
    version="2.4">
     <servlet>
        <servlet-name>usingdoPost</servlet-name>
        <servlet-class>co.usingdoPost</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>usingdoPost</servlet-name>
        <url-pattern>/usingdoPost</url-pattern>
    </servlet-mapping>
    </web-app>
```

将上述 Java 文件进行编译，然后将其复制到包规定的文件下，启动 Tomcat，输入地址 http://localhost:8080/11-8/，运行后会得到如图 11-23 所示的效果。



图 11-23　post 方法封装

11.6　Servlet 过滤器

Servlet 过滤器是 Web 中的一个小型组件，能够拦截来自客户端的请求和响应信息，从而进行查看、提取或者对客户端和服务器之间交换的数据信息进行某项特定的操作。实现的过滤器通常是用来封装一些辅助性的功能方法，这些过滤器方法可能对真正意义上的客户端请求和响应处理不起决定性作用，但还是非常重要的。

11.6.1　过滤器的配置

实现了 Servlet 过滤器之后，还需要通过 web.xml 文件中的两个 XML 元素来声明该过滤器。<filter>元素定义过滤器的名称，并且声明实现类和 init()参数。<filter-mapping>元素将过滤器与 Servlet 或 URL 模式相关联，下面介绍其声明格式。

在<web-app>标记前面的格式如下。

```
<filter>
<filter-name>Page Request Timer</filter-name>
<filter-class>TimeTrackFilter</filter-class>
</filter>
<filter-mapping>
<filter-name>Page Request Timer</filter-name>
<servlet-name>Main Servlet</servlet-name>
</filter-mapping>
<servlet>
<servlet-name>Main Servlet</servlet-name>
<servlet-class>MainServlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>Main Servlet</servlet-name>
<url-pattern>/*</url-pattern>
</servlet-mapping>
```

在 Page Request Timer 过滤器中有两个参数，分别为 page Requst Timer 和 main Servlet，在实际应用中可以通过 config 类中的 getinitparameter()方法获得。

11.6.2　过滤器的应用

在上一节讲解了过滤器的应用流程，这一节将通过一段代码让读者明白过滤器的作用，其代码保存在（光盘：源代码/第 11 章/11-9/）中，具体实现流程如下。

1）新建一个 Java 文件 FilterFlux.java，其代码如下。

```
package com;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
public class FilterFlux extends HttpServlet implements Filter
{
   private static int flux = 0;
public void init(FilterConfig filterConfig) throws ServletException
 {
   }
   public synchronized void doFilter(ServletRequest request, Servlet
Response response,
                     FilterChain filterChain) throws
          ServletException, IOException {
     this.flux++;
     request.setAttribute("flux",String.valueOf(flux));
     filterChain.doFilter(request, response);
   }
```

```
        public void destroy() {
        }
    }
```

2）新建文件 index.jsp，其页面代码如下。

```
<%@ page contentType="text/html; charset=gb2312" language="java"
import="java.sql.*" errorPage="" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>Servlet 过滤器的应用</title>
</head>
<body>
<div align="center">
  <table width="459" height="400" border="0" cellpadding="0"
cellspacing="0" >
    <tr align="center">
      <td ><%=request.getAttribute("flux")%>次</td>
    </tr>
  </table>
  <br>
</div>
</body>
</html>
```

3）新建一个配置文件 web.xml，代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns=http://java.sun.com/xml/ns/j2ee
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schema
Location="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/
j2ee/web-app_2_4.xsd" version="2.4">
  <display-name>web</display-name>
  <filter>
    <filter-name>filterflux</filter-name>
    <filter-class>com.FilterFlux</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>filterflux</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-app>
```

将上述 Java 文件编译，然后将其复制到包规定的文件夹下，启动 Tomcat，输入地址 http://localhost:8080/11-9/，运行后会得到如图 11-24 所示的效果。

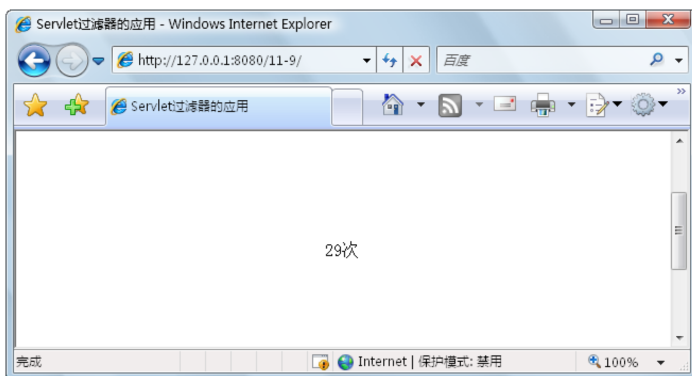


图 11-24　过滤器

11.7　Servlet 监听

Servlet 监听器用于监听一些重要事件的发生，监听器对象可以在事情发生前、发生后做一些必要的处理。在 Servlet 2.4 和 JSP 2.0 中共有 8 个监听器接口和 6 个 Event 类，如表 11-1 所示。

表 11-1　Listener 和 event 类

Listener 接口	Event 类
ServletContextListener	ServletContextEvent
ServletContextAttributeListener	ServletContextAttributeEvent
HttpSessionListener	HttpSessionEvent
HttpSessionActivationListener	
HttpSessionAttributeListener	HttpSessionBindingEvent
HttpSessionBindingListener	
ServletRequestListener	ServletRequestEvent
ServletRequestAttributeListener	ServletRequestAttributeEvent

通过 ServletContext 实例可以存取应用程序的全局对象以及初始化阶段的变量。在 JSP 文件中，通过 JSP 容器默认创建 Application，这是 ServletContext 的实例。在 Servlet 中调用 getServletContext()方法可以得到 ServletContext 的实例。



全局对象即 Application 范围对象，初始化阶段的变量指在 web.xml 中，由<context-param>元素所设定的变量，它的范围也是 Application 范围。

11.8　疑难问题解析

在本章的内容中，详细讲解了 Servlet 的基本知识。本节中，将对本章中比较难理解的

问题进行讲解。

读者疑问：有人听过 Applet，也是网页应用的东西，请问它和 Servlet 有什么相似之处和不同之处？

解答：相同之处：它们不是独立的应用程序，没有 main()方法，它们不是由用户或程序员调用，而是由另外一个应用程序（容器）调用，它们都有一个生存周期，包含 init()和 destroy()方法；不同之处：Applet 具有很好的图形界面（AWT），与浏览器一起，在客户端运行，Servlet 则没有图形界面，运行在服务器端。

读者疑问：Servlet 和 JSP 相比，又有什么不同呢？

解答：JavaServer Pages（JSP）是一种实现普通静态 HTML 和动态 HTML 混合编码的技术，JSP 并没有增加任何本质上不能用 Servlet 实现的功能。但是，在 JSP 中编写静态 HTML 更加方便，不必再用 println 语句来输出每一行 HTML 代码。更重要的是，借助内容和外观的分离，页面制作中不同性质的任务可以方便地分开，比如，由页面设计者进行 HTML 设计，同时留出供 Servlet 程序员插入动态内容的空间。



职场点拨——寻兼职

在生活中都会遇到一个同样的问题，无论是买房、买车、旅行散心，还是和三五知己一块聚餐，都需要钱。程序员也是凡夫俗子，也需赚钱来养家糊口。固定的那些工资有时满足不了我们花钱的欲望，怎么办？寻找兼职干外快是解决上述问题的一个很好途径。现在 IT 行业外包盛行，很适合发展兼职。程序员们可以通过以下渠道获取私活。

1）客户：你曾经为这个客户服务过，你已经熟悉了他的性格，他也认同了你的技术。此时你们就有很大的合作机会。

优点是：因为以前曾经合作过，所以在流程上会很顺利；缺点是：他认识你同事或上司，可能不经意间泄露你们的私活，对你的本职会有影响。

2）朋友或同学介绍：在日常生活中，朋友和同学知道你的职业，他们会很自然地为你介绍一些客户。

优点是：对方肯定真实可靠；缺点是：可能碍于面子，收入可能会有影响，在日后服务上可能会耗费时间。

3）网络资源：当前互联网迅猛发展，各个私活站点如雨后春笋般纷纷建立。我们可以登录一些主流网站的私活板块，来寻找自己的私活。例如，威客、365huo、CSDN 等，都是主流的私活交易平台。

优点是：信息量大；缺点是：竞争力大，信任度较低。

4）开一个工作室：如果时间足够，可以叫上三五知己或同学，合伙开一个工作室。

优点是：有正规的办公地点，更能吸引一些客户；缺点是：需要投入人力、物力和财力。

第 12 章　JavaBean 组件技术

JavaBean 是一种用 Java 语言写成的可重用组件。书写的类必须是具体的和公共的，并且具有无参数的构造器。JavaBean 就是 Java 类，属于特定的重要组件，并且扩展了适应性和范围，允许用户访问内部的属性和方法。通过本章能学到如下知识。

- JavaBean 是什么。
- 设置 JavaBean 属性。
- JavaBean 方法。
- JavaBean 的作用范围。
- JavaBean 的移除。
- 职场点拨——兼职可靠吗？

2010 年 XX 月 X 日，阴

原来在网络上可以找到这么多的兼职信息啊，经过一上午的在线沟通后，我终于达成了一个合作意向。对方是外地的，只能通过 QQ 洽谈业务了。但是转念一想，人都没见着，可靠吗？



一问一答

小菜：“我在网上找到了一个兼职，但是对方是外地的，我们只能在网上洽谈业务，这可靠吗？”

Wisdom: “接兼职是有技巧的，不是随便可以接的，我们需要验证可靠性。在本章最后介绍了兼职的处理方式，你可以借鉴一下！”

小菜：“哎，还是学习吧，本章的 JavaBean 是什么？”

Wisdom: “JavaBean 是一个优秀的可重用组件，通过它，用户省下很多代码，在维护程序上也会省下许多事情。例如，可以通过 JavaBean 封装注册登录中的用户信息和一些逻辑处理。通过 JavaBean 可以提高代码的重用性，有利于页面设计人员维护 HTML。”

12.1　认识 JavaBean

JavaBean 是一个优秀的可重用组件，可以为开发人员省下很多代码。在下面的代码中，使用 JavaBean 封装了注册登录应用中的用户信息和逻辑处理。代码保存在（光盘：源代码/

第 12 章/12-1/）中，具体实现流程如下。

1）新建文件 test.java，其代码如下。

```
package test;
public class Test
{
private String name=null;
public Test(){
}
public void setName(String name)
{
      this.name=name;
}
public String getName()
{
      return name;
}
}
```

2）新建文件 bean1.jsp，其实现代码如下。

```
<%@ page contentType="text/html; charset=gb2312" import="test.Test" %>
<html>
<body>
<center>
<font color="blue" size="6"><br><br>
<%
Test bean = new Test();
bean.setName("first javabean tested success!");
%>
<%= bean.getName() %>
</font>
</center>
</body>
</html>
```

3）新建文件 bean2.jsp，其实现代码如下。

```
<%@ page contentType="text/html; charset=gb2312" %>
<%@ page import="test.Test" %>
<jsp:useBean id="bean" class="test.Test" scope="page"/>
<html>
<body>
<center>
<font color="red" size="6">
<%
bean.setName("second bean succeded!");
%>
```

```
        <%= bean.getName() %>

        </center>
        </body>
        </html>
```

4）新建文件 bean3.jsp，其实现代码如下。

```
        <%@ page contentType="text/html; charset=gb2312" import="test.Test" %>
        <jsp:useBean id="bean" class="test.Test" scope="page"/>
        <html>
        <body>
        <center>
        <font color="brown" size="6">
        <jsp:setProperty name="bean" property="name" value="third bean tested
succeeded!!"/>
        <jsp:getProperty name="bean" property="name"/>
        </center>
        </body>
        </html>
```

将上述 Java 源代码编译并将代码文件保存到服务器的环境下，浏览第 1 个页面得到如图 12-1 所示的效果。

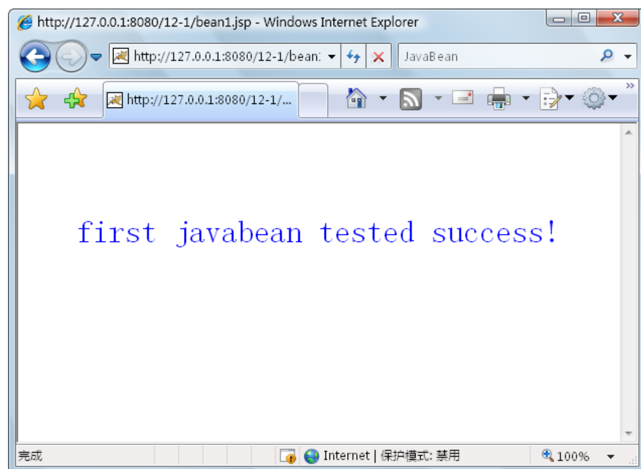


图 12-1　javabean 的体验

然后浏览第 2 个页面，效果如图 12-2 所示。

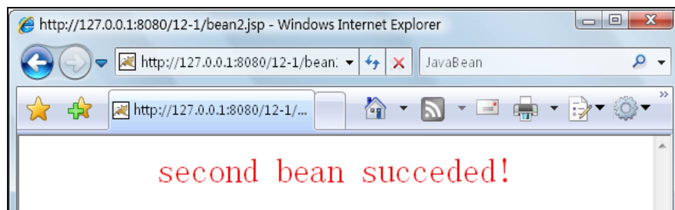


图 12-2　第 2 个页面

在浏览器中浏览第 3 个页面，效果如图 12-3 所示。

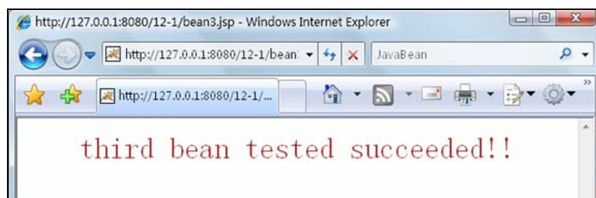


图 12-3　第 3 个页面

12.2　JavaBean 基础

JavaBean 是描述 Java 的组件模型，类似于 Microsoft 中的 COM 组件。开发人员可以使用 JavaBean 将功能、处理、值、数据库访问和其他任何可以用 Java 代码创造的对象进行打包，并且其他的开发者可以通过内部的 JSP 页面、Servlet、其他 JavaBean、Applet 程序或者应用来使用这些对象。

12.2.1　JavaBean 介绍

推出 JavaBean 的最初的目的是将可以重复使用的软件代码打包，帮助厂家开发在综合开发环境（IDE）下使用的 Java 软件部件。例如，这些包有 Grid 控件，用户可以将该部件拖放到开发环境中。从此，JavaBean 就可以扩展为一个 Java Web 应用的标准部件，并且 JavaBean 部件框架已经扩展为企业版的 Bean（EJB）。

JavaBean 就是 Java 类，并且扩展了适应性和范围，允许用户访问内部的属性和方法。JavaBean 类可以用于下列方法。

- ❑ 在 IDE 中，即使 JavaBean 是被编译的，并且无法利用原始的源文件，JavaBean 也允许应用开发者浏览其中的方法。
- ❑ 在 Remote Method Invocation（RMI）的分布式风格中，此功能还不吸引开发人员的眼球，但是在包含企业版的 JavaBean 中将改变这种情况。
- ❑ JavaBean 的属性值和状态可以保存到磁盘上。

12.2.2　体验 JavaBean

JavaBean 能够在不同的领域表现出卓越的性能，JavaBean 可分为两种，分别是可视化和非可视化。可视化的 JavaBean 具有 GUI 图形用户界面，非可视化的 JavaBean 是没有 GUI 界面的。在 JSP 程序中，通常用 JavaBean 封装事务逻辑、数据库操作等，这样可以很好地实现业务逻辑与前台程序（如 jsp 文件）的分离，使得系统具有更好的健壮性和灵活性。例如，在购物车程序中，要实现在购物车中添加一件商品这个功能，就可以写一个购物车操作的 JavaBean，建立一个 public 的 AddItem()成员方法，在前台 JSP 文件里面直接调用这个方法即可实现。如果后来又考虑添加商品的时候需要判断库存是否有货物，没有货物不得购买，此时可以直接修改 JavaBean 的 AddItem()方法，加入处理语句来实现，这样就完全不用修改前台 JSP 程序了，下面通过一个例子来体验 JavaBean 的用处。

实例 31：在局部范围内访问全局变量

在局部范围内访问全局变量，需要使用一个关键字来实现。下面通过一段代码来说明，其代码（光盘：源代码/第 12 章/ FirstJavaBean.java）如下。

```
import java.io.*;
public class FirstJavaBean {
private String FirstProperty = new String("");
public FirstJavaBean()
{ //构造方法
}
public String getFirstProperty()//get***方法
{
return FirstProperty;
}
public void setFirstProperty(String value)
{
FirstProperty = value;
}
public static void main(String[] args)
{
System.out.println("大家好，第一个 JavaBean 跟大家打个招呼");
}
}
```

将上述代码进行编译并运行，得到如图 12-4 所示的效果。

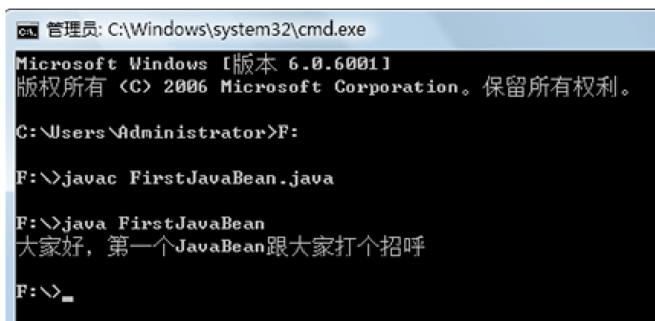


图 12-4　第一个 JavaBean


多学一招

在上面实例中，创建了一个简单的 JavaBean 程序。再看下面一段 JavaBean 代码，让读者更加熟悉创建 Javabean 的流程，其代码（光盘：源代码/第 12 章/SimpleJavaBean. Java）如下。

```
public class SimpleJavaBean implements java.io.Serializable
{
   public SimpleJavaBean()
```

```
    {
            }
    private String id;
    private String name;
    private String password;
    public String getId()
    {
        return id;
    }
    public void setId(String id)
    {
        this.id = id;
    }
    public String getName()
    {
        return name;
    }
    public void setName(String name)
    {
        this.name = name;
    }
    public String getPassword()
    {
        return password;
    }
    public void setPassword(String password)
    {
        this.password = password;
    }
}
```

提示：在 JSP 中使用 JavaBean 组件，在没有实现 Java.io.serializable 接口的情况下，仍然可以运行 JavaBean 组件，并且 JavaBean 必须是一个 Public 类。

12.2.3　在 JSP 中调用 JavaBean

Java Web 是 Java 开发的强项，初学者应该尽快掌握在 Web 页面中是如何调用 JavaBean 组件的。下面将通过几段代码来讲解在 JSP 中调用 JavaBean 的方法，代码保存在（光盘：源代码/第 12 章/12-2/）中，具体实现流程如下。

1）新建文件 ChartGraphics.java，其实现代码如下。

```
package com;
import java.io.*;
import java.util.*;
```

```
        import com.sun.image.codec.jpeg.*;
        import java.awt.image.*;
        import java.awt.*;
        public class ChartGraphics
        {
        BufferedImage image;
        public void createImage(String fileLocation)
        {
          try {
           FileOutputStream fos = new FileOutputStream(fileLocation);
           BufferedOutputStream bos = new BufferedOutputStream(fos);
           JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(bos);
           encoder.encode(image);
           bos.close();
          } catch(Exception e) {
           System.out.println(e);
          }
        }
        public void graphicsGeneration(int h1,int h2,int h3,int h4,int h5)
        {
          final int X=10;
          int imageWidth = 300;//图片的宽度
          int imageHeight = 300;//图片的高度
          int columnWidth=30;//柱的宽度
          int columnHeight=200;//柱的最大高度
          ChartGraphics chartGraphics = new ChartGraphics();
          chartGraphics.image  =  new  BufferedImage(imageWidth,  imageHeight,
BufferedImage.TYPE_INT_RGB);
          Graphics graphics = chartGraphics.image.getGraphics();
          graphics.setColor(Color.white);
          graphics.fillRect(0,0,imageWidth,imageHeight);
          graphics.setColor(Color.red);
          graphics.drawRect(X+1*columnWidth, columnHeight-h1, columnWidth, h1);
          graphics.drawRect(X+2*columnWidth, columnHeight-h2, columnWidth, h2);
          graphics.drawRect(X+3*columnWidth, columnHeight-h3, columnWidth, h3);
          graphics.drawRect(X+4*columnWidth, columnHeight-h4, columnWidth, h4);
          graphics.drawRect(X+5*columnWidth, columnHeight-h5, columnWidth, h5);
          chartGraphics.createImage("D:\\temp\\chart.jpg");
        }
        public  static void main(String[] args){
          ChartGraphics bb=new ChartGraphics();
          bb.graphicsGeneration(10, 20, 30, 45, 60);
         }
        }
```

2）新建文件 GetData.java，其实现代码如下。

```
package com;
import java.io.*;
public class GetData
 {
int heightArray[] = new int[5];
public int[] getHightArray()
{
try {
RandomAccessFile randomAccessFile = new RandomAccessFile ("d:\\temp\\
ColumnHeightArray.txt","r");
for (int i=0;i<5;i++ )
{
heightArray[i] = Integer.parseInt(randomAccessFile.readLine());
}
}
catch(Exception e) {
System.out.println(e);
}
return heightArray;
}
}
```

3）新建文件 myjsp.jsp，其实现代码如下。

```
<%@ page import="com.ChartGraphics" %>
<%@ page import="com.GetData" %>
<jsp:useBean id="cg" class="com.ChartGraphics"/>
<jsp:useBean id="gd" class="com.GetData"/>
<%!
int height[]=new int[5];
%>
<%
height=gd.getHightArray();
cg.graphicsGeneration(height[0],height[1],height[2],height[3],height[4]);
%>
<html>
<body>
<img src="chat.jpg"></img>
</body>
</html>
```

将上述代码进行编译并运行，运行之后即可使用 JavaBean 实现功能，执行效果如图 12-5 所示。

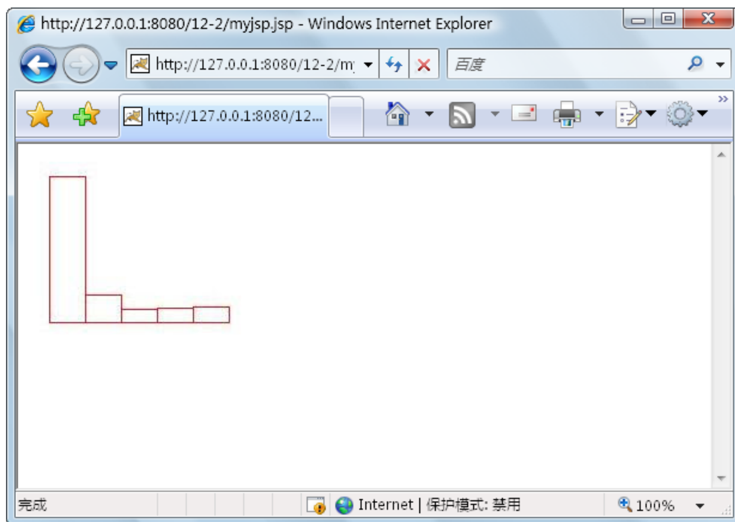


图 12-5　在 JavaWeb 中使用 JavaBean

12.3　设置 JavaBean 属性

JavaBean 的属性与一般 Java 程序中所指的属性是一个概念，在程序中的具体体现就是类中的变量。在 JavaBean 设计中，按照属性的不同作用又可以细分为 Simple, Index, Bound 与 Constrained 4 类。在本节的内容中，将详细讲解设置 JavaBean 属性的知识。

12.3.1　Simple 属性

Simple（简单）属性可以表示一个伴随有一对 get/set 方法（C 语言的过程或函数在 Java 程序中称为“方法”）的变量。属性名与和该属性相关的 get/set 方法名对应。例如，如果有 setX 和 getX 方法，则暗指有一个名为“X”的属性。如果有一个方法名为 isX，则通常暗指“X”是一个布尔属性（即 X 的值为 true 或 false）。看下面的代码。

```
public class alden1 extends Canvas
{
string ourString= "Hello";
//属性名为 ourString，类型为字符串
public alden1(){
//alden1()是 alden1 的构造函数。
setBackground(Color.red);
setForeground(Color.blue);
}
/* "set"属性*/
public void setString(String newString)
 {
ourString=newString;
}
```

```
/* "get"属性 */
public String getString()
{
return ourString;
}
}
```

如果简单的数据的类型为 boolean，除了可以使用 setxxx()方法外，还可以使用 isXXX 方法代替 getXXX()方法，其代码如下。

```
Public boolean isFlog()
{
Return flag;
}
Public void setFlog(boolean flag)
{
This.flag=flag;
}
```

12.3.2 Indexed 属性

一个 Indexed 属性表示一个数组值。使用与该属性对应的 set/get 方法可取得数组中的数值。该属性也可一次设置或取得整个数组的值，其代码如下。

```
public class alden2 extends Canvas
{
int[] dataSet={1,2,3,4,5,6}; // dataSet 是一个 indexed 属性
public alden2()
{
setBackground(Color.red);
setForeground(Color.blue);
}
/* 设置整个数组  */
public void setDataSet(int[] x)
{
dataSet=x;
}
/* 设置数组中的单个元素值 */
public void setDataSet(int index, int x)
{
dataSet[index]=x;
}
/* 取得整个数组值 */
public int[] getDataSet()
{
return dataSet;
}
```

```
        /* 取得数组中的指定元素值 */
        public int getDataSet(int x)
        {
        return dataSet[x];
        }
        }
```

12.3.3　Bound 属性

一个 Bound 属性是指当该种属性的值发生变化时要通知其他对象，当每次属性值改变时，这种属性激活一个 PropertyChange 事件（在 Java 程序中，事件也是一个对象）。在事件中通常封装了属性名、属性的原值、属性变化后的新值。将事件传递到其他的 Bean，至于接收后 Bean 会做什么 Bound 属性并不负责。为了说明上述 Bound 属性的作用，请看下面的代码。

```
        public class alden3 extends Canvas
        {
        String ourString= "Hello";
        //ourString 是一个 bound 属性
        private PropertyChangeSupport changes = new PropertyChange Support
(this); /** 注：因为 Java 是纯面向对象的语言，
        如果要使用某种方法则必须指明是要使用哪个对象的方法，
        在下面的程序中要进行激活事件的操作，
        这种操作所使用的方法是在 PropertyChangeSupport 类中的。
        所以上面声明并实例化了一个 changes 对象，
        在下面将使用 changes 的 firePropertyChange 方法来点火 ourString 的属性改变事件。*/
        public void setString(string newString)
        {
        String oldString = ourString;
        ourString = newString;
        /* ourString 的属性值已发生变化，于是接着点火属性改变事件 */
        changes.firePropertyChange("ourString",oldString,newString);
        }
        public String getString()
        {
        return ourString;
        }
        /** 以下代码是为开发工具所使用的。
        不能预知 alden3 将与哪些 Beans 组合成为一个应用，
        无法预知如果 alden3 的 ourString 属性发生变化时有哪些组件与此变化有关，
        因而 alden3 这个 Beans 要预留出一些接口给开发工具，开发工具使用这些接口，把其他的
JavaBeans 对象与 alden3 挂接。*/
        public void addPropertyChangeListener(PropertyChangeLisener l)
        {
        changes.addPropertyChangeListener(l);
        }
```

```
        public void removePropertyChangeListener(PropertyChangeListener l)
        {
        changes.removePropertyChangeListener(l);
        }
```

在上面的代码中，开发工具调用 changes 的 addPropertyChangeListener()方法 ，把其他 JavaBean 注册到 ourString 属性的监听者队列 l 中。队列 l 是一个 Vector 数组，可存储任何 Java 对象。开发工具也可使用 changes 的 removePropertyChangeListener()方法，从队列 l 中注销指定的对象，使 alden3 的 ourString 属性的改变不再与这个对象有关。当开发人员手写代码编程时，也可直接调用这两个方法，把其他 Java 对象与 alden3 挂接。

12.3.4 Constrained 属性

JavaBean 中的 Constrained 属性是指当这个属性的值要发生变化时，与这个属性已建立了某种连接的其他 Java 对象可以否决属性值的改变。Constrained 属性的监听者通过抛出 PropertyVetoException 的方法来阻止该属性值的改变。下面通过一段代码来讲解 Constrained 属性的用法，其代码如下。

```
        public class JellyBean extends Canvas
        {
        private PropertyChangeSupport changes=new PropertyChangeSupport(this);
         private VetoableChangeSupport Vetos=new VetoableChangeSupport(this);
         /*与前述 changes 相同，可使用 VetoableChangeSupport 对象的实例 Vetos 中的方法，
在特定条件下来阻止 PriceInCents 值的改变。*/
        ......
         public void setPriceInCents(int newPriceInCents)
        throws PropertyVetoException
        {
        /* 方法名中 throws PropertyVetoException 的作用是当有其他 Java 对象否决 Price
InCents 的改变时，要抛出例外。*/
         /* 先保存原来的属性值*/
        int oldPriceInCents=ourPriceInCents;
         /*点火属性改变否决事件*/
vetos.fireVetoableChange("priceInCents",new    Integer(OldPriceInCents),    new
Integer(newPriceInCents));
         /*若有其他对象否决 priceInCents 的改变，则程序抛出例外，不再继续执行下面的两条语
句，方法结束。若无其他对象否决 priceInCents 的改变，则在下面的代码中把 ourPriceIncents 赋
予新值，并让属性改变事件*/
        ourPriceInCents=newPriceInCents;
         changes.firePropertyChange("priceInCents", new Integer
        (oldPriceInCents),new Integer(newPriceInCents));
         }
        /*与前述 changes 相同，也要为 PriceInCents 属性预留接口，使其他对象可注册入
PriceInCents 否决改变监听者队列中，或把该对象从中注销*/
        public void addVetoableChangeListener(VetoableChangeListener l)
         {
```

```
vetos.addVetoableChangeListener(l);
 }
public void removeVetoableChangeListener(VetoableChangeListener l)
{
vetos.removeVetoableChangeListener(l);
 }
......
 }
```

从上面的例子中可看到，一个 Constrained 属性有属性变化监听者和否决属性改变的监听者两种监听者。否决属性改变的监听者在自己的对象代码中有相应的控制语句，在监听到有 constrained 属性要发生变化时，在控制语句中判断是否应否决这个属性值的改变。总之，某个 Bean 的 Constrained 属性值可否改变取决于其他的 Bean 或者是 Java 对象是否允许这种改变。允许与否的条件由其他的 Bean 或 Java 对象在自己的类中进行定义。

12.4　JavaBean 方法

通过本书前面章节的学习，了解了对象中有属性和方法两种元素，其中方法可以利用属性特征去实现一些功能。在本节的内容中，将通过一个具体实例来讲解使用 JavaBean 方法的流程。

实例 32：JavaBean 方法的应用

本实例演示了使用 JavaBean 方法的流程，其代码（光盘：源代码/第 12 章/JavaBeanmethod.java）如下。

```
import java.util.HashMap;
import java.util.Map;
public class JavaBeanmethod
{
   private String[]array=new String[6];
   private Map map=new HashMap();//带有 KEY 值
   public Map getMap()
{
         return this.map;
   }
   public Object getMap(Object key)
{
         return map.get(key);
   }
   public void setMap(Object value,Object key)
{
         map.put(value, key);
   }
   public void printMap(){
         for(int i=0,j=10;i<10;i++,j--)
```

```
        {
                    setMap(i+" ",(i*j)+" ");
                    if(map.containsKey(i+" "))
        {
                         System.out.println("map{key="+i+""+"value="+map.get
(i+" ")+"}");
                    }
               }
          }
          public static void main(String args[]){
               JavaBeanmethod simple=new JavaBeanmethod();
               simple.printMap();
          }
          public void setMap(Map map)
        {
               this.map = map;
          }
          public void setArray(int index,String value)
        {
               this.array[index]=value;
          }
          public String getArray(int index)
        {
               return array[index];
          }
        }
```

编译运行后得到如图 12-6 所示的效果。

```
管理员: C:\Windows\system32\cmd.exe

F:\>java JavaBeanmethod
map{key=0value=0 }
map{key=1value=9 }
map{key=2value=16 }
map{key=3value=21 }
map{key=4value=24 }
map{key=5value=25 }
map{key=6value=24 }
map{key=7value=21 }
map{key=8value=16 }
map{key=9value=9 }

F:\>_
```

图 12-6　执行的结果



多学一招

通过上述实例，读者领会了 JavaBean 的方法的作用。在下面的代码中，展示了一个关于数据库操作 JavaBean 的流程，由于本书还没有讲解数据库的知识，读者只需更好地理解

JavaBean 的优点即可，而无需考虑数据库的知识。其代码【光盘：源代码/第 12 章/jdbc_sql.java】如下。

```
import java.sql.*;
import java.util.*;
import java.text.*;
import java.io.*;
public class jdbc_sql
{
String conn_str="jdbc:mysql: //localhost/HeaVeN_news";//---------数据库
String username="root";      //---------------数据库用户名
String password="";          //-------------数据库密码
private Connection conn=null;
private Statement stmt=null;
ResultSet rs=null;
//----------------------数据库链接----------------
public jdbc_sql()
{
try{
Class.forName("org.gjt.mm.mysql.Driver");
}
catch(java.lang.ClassNotFoundException e)
{
System.err.println("dbclass():"+e.getMessage());
}
try{
conn=DriverManager.getConnection(conn_str,username,password);
}
catch(SQLException ex) {
System.err.println("dbclass():"+ex.getMessage());
}
}
//---------------------数据库查询-----------------
public ResultSet query(String sql)
{
try{
stmt=conn.createStatement();
rs=stmt.executeQuery(sql);}
catch(SQLException ex) {
System.err.println("query():"+ex.getMessage());
}
return rs;
}
//————————————数据库修改—————————
public void update(String sql)
{
```

```
        try{
        stmt=conn.createStatement();
        stmt.executeUpdate(sql);}
        catch(SQLException ex) {
        System.err.println("update():"+ex.getMessage());
        }
        }
        //———————————关闭数据源———————
        public void closestmt()
        {
        try{stmt.close();
        }
        catch(SQLException ex) {
        System.err.println("closestmt():"+ex.getMessage());
        }
        }
        //———————————关闭链接—————————
        public void closeconn()
        {try{conn.close();}
        catch(SQLException ex) {System.err.println("closeconn()"+ex.get
Message());}
        }
        //---------------------转换中文
        public String toChinese(String s){
        if(s==null)
        return null;
        try{
        String convert=new String(s.getBytes("gb2312"),"ISO8859-1");
        return convert;
        }catch(Exception e){}
        return null;
        }
        //--------------------分页
        public  String  page1(int  num,int  line,int  page_line,int  start,int
page,String search,String filename){
        int pagetotal=line*page_line;                //所要显示的总条数
        int pagestart;//
        int pagenum;//
        int total=(num-start+line-1)/line;           //取得总页数
        int total_start=(num+pagetotal-1)/pagetotal; //取得总偏移数
        String returnstr="";
        if(start!=0)
        {
        int old_start=start-pagetotal;
        returnstr+=" <a href="+filename+"?page=0&start="+old_start+search+">
<<</a>";
```

```
        }
        pagestart=start/pagetotal*page_line;          //取得当前页数
        for(int i=0;i<total&&i<page_line;i++)
        {
        pagenum=pagestart+i+1;
        returnstr+="  <a  href="+filename+"?page="+i+"&start="+start+search+">"
+pagenum+"</a>";
        }
        if(total_start!=(start/pagetotal+1)&&num!=0)
        {
        int new_start=start+pagetotal;
        returnstr+=" <a href="+filename+"?page=0&start="+new_start+search+">>>
</a>";
        }
        return returnstr;
        }

        //--------------------------------取得时间
        public String GetPath(){
        String dateString="123";
        try{
        java.text.SimpleDateFormat  formatter=new  java.text.SimpleDate  Format
("yyyy-MM");
        java.util.Date currentTime_1=new java.util.Date();
        dateString=formatter.format(currentTime_1);
        }catch(Exception e){
        }
        return dateString;
        }

        //--------------------------------取得时间
        public String strNowtime(){
        String dateString="123";
        try{
        java.text.SimpleDateFormat   formatter=new   java.text.SimpleDateFormat
("yyyy-MM-dd hh:mm:ss");
        java.util.Date currentTime_1=new java.util.Date();
        dateString=formatter.format(currentTime_1);
        }catch(Exception e){
        }
        return dateString;
        }

        //-------------------------去掉 html--------------------------
        public String HTMLEncode(String Str){
        Str=ebbReplace(Str,"<","<");
```

```
        Str=ebbReplace(Str,">",">");
        Str=ebbReplace(Str," ","<BR>");
        return Str;
        }
        //-------------------------替换变量--------------------------
        public String ebbReplace(String Str,String oldStr,String newStr){
        String ReturnStr="";
        int i,j,t,m,n;
        n=0;
        j=oldStr.length();
        if (Str.indexOf(oldStr)>-1)
        {
        while(Str.indexOf(oldStr,n)>-1)
        {
        i=Str.length();
        if (Str.indexOf(oldStr)==0)
        Str=newStr+Str.substring(j,i);
        else
        {
        t=Str.indexOf(oldStr);
        m=(t+j);
        Str=Str.substring(0,t)+newStr+Str.substring(m,i);
        n=t+newStr.length()-j+1;
        }
        }
        }
        ReturnStr=Str;
        return ReturnStr;

        }
        //---------------------------ubb 替换-------------------------
        public  String  LCReplace(String  Str,String  BStr,String  EStr,String
ReStr){
        String ReturnStr="",Str1="",Str2="";
        int i,j,n;
        n=0;
        if ((Str.indexOf(BStr)>-1)&&((Str.indexOf(EStr)>-1)))
        {
        while(Str.indexOf(BStr,n)>-1)
        {
        i=Str.indexOf(BStr);
        j=Str.indexOf(EStr);
        Str1=Str.substring((i+BStr.length()),j);
        Str2=ebbReplace(ReStr,"$lichao$",Str1);
        Str1=BStr+Str1+EStr;
```

```
        Str=ebbReplace(Str,Str1,Str2);
        n=i+Str2.length()-Str1.length();
        }
        }
        ReturnStr=Str;
        return ReturnStr;
        }
        }
```

在上述代码中，使用了一个运行在数据库上的 JavaBean，读者只有在学习完本书的第三篇内容后，才能读懂上述代码。此时读者只看结构和功能即可，不必完全弄懂上述代码的真正含义。

12.5　JavaBean 的作用范围

JSP 作为一个很好的动态网站开发技术，逐渐得到了越来越广泛的应用。在各类 JSP 应用程序中，JSP+JavaBean 的组合成为了最常见的 JSP 开发标准。在本节的内容中，让读者体会 JSP 是如何与 JavaBean 结合在一起的。

12.5.1　Page 作用域

如果设定一个 JSP 页面为作用域，通常使用如下代码实现。

```
<jsp:useBean id="bean" class="com.CountBean" scope="page"/>
```

这个作用域是最严格的。页面域的对象只对于它所在页面是可访问的。页面域的 JavaBean 和脚本程序创建的对象都是线程安全的。

12.5.2　Request 的作用域

请求域的对象只在相应请求的生命周期内有效，即对象在创建它的页面内有效，以及请求被转发或者被包含的页面内有效。请求域的对象是线程安全的，只有相应请求的执行线程可以访问这些对象，其格式如下。

```
<jsp:useBean id="bean" class="com.CountBean" scope="request"/>
```

12.5.3　Session 的作用域

会话域的对象在参与某个客户端会话的所有应用组件时是有效的。会话域的对象不是线程安全的，如果有多个请求同时访问同一个会话（Session）对象，那么需要对此对象进行同步访问，其格式如下。

```
<jsp:useBean id="bean" class="com.CountBean" scope="session"/>
```

12.5.4　Application 的作用域

这是最宽泛的一种作用域，这种应用域对象在应用的整个生命周期内都是有效的。

Application 作用域对象不是线程安全的，如果用多个请求来同时修改某一个对象，那么必须对这些访问进行同步处理，其格式如下。

```
<jsp:useBean id="bean" class="com.CountBean" scope="application"/>
```

12.6　JavaBean 的移除

当使用完 JavaBean 后应该立即释放空间，使其不再继续占用资源，这样内存可以去处理其他的信息。移除 JavaBean 的方法十分简单，只需要输入如下语句即可实现。

```
Pagecontex.removeAttribute(String name);
Request.removeAttibute(String name);
```

接下来将通过几段代码讲解如何释放内存。代码保存在（光盘：源代码/第 12 章/12-4/）中，具体实现流程如下。

1）文件 myBean.java 的代码如下。

```
public class myBean {
   private String name;
   private String password;
   private int count=0;
   public int getCount() {
         count++;
         return count;}
   public void setCount(int count) {
               this.count = count;}
   public String getName() {
         StringTrans s=new StringTrans();
         return s.tranC(name);}
   public void setName(String name) {
         this.name = name;}
   public String getPassword() {
         return password;}
   public void setPassword(String password) {
         this.password = password;
   }}
```

2）文件 StringTrans.java 的代码如下。

```
import java.io.*;
public class StringTrans{
  public static String tranA(String chi) {
    String result=null;
    byte temp[];
    try{
      temp=chi.getBytes("iso-8859-1");
```

```
            result=new String(temp);}
        catch(UnsupportedEncodingException e){
            System.out.println(e.toString());}
          return result;}
        public static String tranB(String chB){
          String result=null;
          byte temp[];
          try{
            temp=chB.getBytes("UTF-8");
            result=new String(temp,"iso-8859-1");}
        catch(UnsupportedEncodingException e){
            System.out.println(e.toString());}
          return result;}
          public static String tranC(String chB){
          String result=null;
          byte temp[];
          try {
            temp=chB.getBytes("iso-8859-1");
            result=new String(temp,"UTF-8");}
        catch(UnsupportedEncodingException e) {
            System.out.println(e.toString()); }
          return result;
        }}
```

3）文件 Login.jsp 的代码如下。

```
        <%@ page contentType="text/html; charset=utf-8" language="java" import
="java.sql.*" errorPage="" %>
        <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
        "http://www.w3.org/TR/html4/loose.dtd">
        <jsp:useBean id="myBean" scope="session" class="com.wsy.myBean"/>
        <html>
        <head>
        <script language="javascript">
        function check(){
           if(document.all.name.value.length==0){
                alert("请填写用户名!");
                return false;
           }
           if(document.all.password.value.length==0){
                alert("请填写密码!");
                return false;
           }
           form1.submit();
           return true;
        }
        function reset(){
```

```
            document.all.form1.reset();
            }
        </script>
        <title>论坛登录</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <style type="text/css">
        <!--
        .style2 {
            font-size: 18px;
            color: #999999;
        }
        .style3 {color: #999999}
        .style4 {font-size: 12px}
        -->
        </style>
        </head>
        <body  bgcolor="#FFFFFF"  leftmargin="0"  topmargin="0"  marginwidth="0"
marginheight="0">
        <!-- ImageReady Slices (登录 2.jpg) -->
        <h2 align="center" class="style2">JavaBean 的移除</h2>
        <table width="407" height="226" border="0" align="center" cellpadding=
"0" cellspacing="0" id="__01">
        <form action="login.jsp" name="form1">
            <tr>
                 <td colspan="6">
                      <img src="images/01.gif" width="407" height="60" alt="">
</td>
            </tr>
            <tr>
                 <td rowspan="4">
                      <img  src="images/02.gif"  width="130"  height="137"  alt=
""></td>
                 <td>
                      <img src="images/03.gif" width="56" height="47" alt="">
</td>
                 <td    colspan="3"    width="117"    height="47"    background
="images/04.gif"><input name="name" type="text" size="10">
                      <!--img src="images/04.gif" width="117" height="47" alt=
""--></td>
                 <td>
                      <img src="images/05.gif" width="104" height="47" alt="">
</td>
            </tr>
            <tr>
                 <td>
                      <img src="images/06.gif" width="56" height="43" alt="">
```

```
</td>
                <td colspan="3" width="117" height="43"><input name="password"
type="password" size="11">
                     <!--img src="images/07.gif" width="117" height="43" alt=
""--></td>
                <td>
                     <img src="images/08.gif" width="104" height="43" alt="">
</td>
           </tr>
           <tr>
                <td rowspan="2">
                     <img src="images/09.gif" width="56" height="47" alt="">
</td>
                <td>
                     <img src="images/10.gif" alt="" width="52" height="25"
border="0" usemap="#Map" ></td>
                <td>
                     <img src="images/11.gif" width="9" height="25" alt="">
</td>
                <td>
                     <img src="images/12.gif" alt="" width="56" height="25"
border="0" usemap="#Map2"></td>
                <td>
                     <img src="images/13.gif" width="104" height="25" alt="">
</td>
           </tr>
           <tr>
                <td colspan="4">
                     <img src="images/14.gif" width="221" height="22" alt="">
</td>
           </tr>
           <tr>
                <td colspan="6">
                     <img src="images/15.gif" width="407" height="29" alt="">
</td>
           </tr>
        </table>
        <!-- End ImageReady Slices -->
        <div align="center">
          <map name="Map">
            <area shape="rect" coords="1,1,51,25" href="#" onClick="return
check();">
          </map>
          <map name="Map2">
            <area shape="rect" coords="2,1,52,25" href="#" onClick="reset();">
          </map>
```

```
        <%if(request.getParameter("name")!=null){ %>
        <span class="style3">
        <jsp:setProperty name="myBean" property="*"/>
        <span class="style4">用户名为
        <jsp:getProperty name="myBean" property="name"/>
        登录
        <jsp:getProperty name="myBean" property="count"/>
        次</span></span>
        <%
        session.removeAttribute(myBean.getCount()+"");}
        %><br>
      </div>
      </body>
      </html>
```

将上述 Java 文件进行编译并运行，得到如图 12-7 所示的效果。

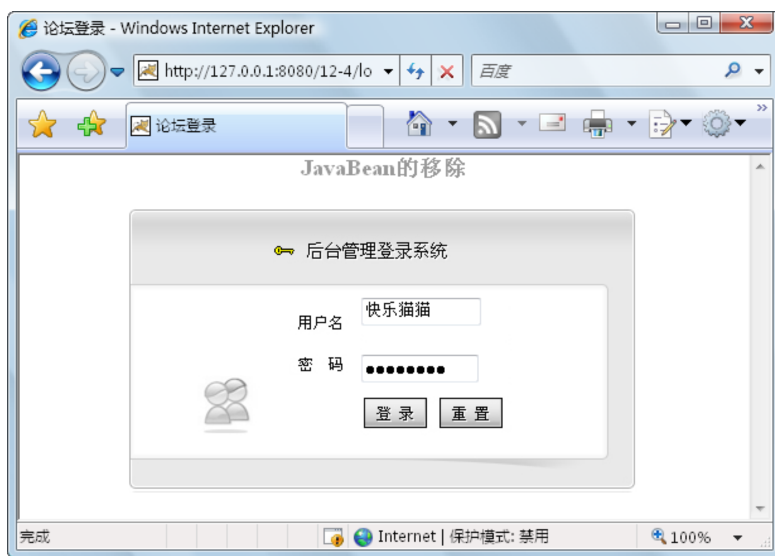


图 12-7 JavaBean 的移除

12.7 疑难问题解析

在本章的内容中，详细讲解了 JavaBean 组件技术的基本知识。本节中，将对本章中比较难理解的问题进行讲解。

读者疑问：JSP+JavaBean 到底是什么样的模式？JavaBean 中不也是定义类和方法的嘛，使用 JavaBean 和直接在 JSP 页面中调用包中相应类中的方法有什么区别？

解答：这很好理解，使用之后，在 JSP 中不会再有大量的 Java 代码，前台后台分工更加明确，编写程序更容易，维护也更加容易。

读者疑问： J2EE 中 JavaBean 到底是什么？表现层？逻辑处理层？还是数据层？JavaBean 是属于哪个层次的？

解答： JavaBean 是一种组件技术，是数据层。对应 Microsoft 的 COM 技术。JSP 是 Java Server Page 的缩写，能够在 C/S 和 B/S 结构中实现 Server 端编程，和 Microsoft 的 ASP 类似。JDBC 是一种 Java 与数据库的驱动技术，对应 Microsoft 的 ODBC。如果想从事 Web 开发，上述技术都要学习，尤其是 JavaBean，如果将它们组合起来就是所谓的 MVC 的开发模式，在本书后面的内容中，将讲解 JDBC 数据库编程的基本知识。



职场点拨——兼职可靠吗？

上一章中介绍了寻找兼职的常用方法和渠道，为了保证我们在干完兼职之后能获得相应的收益，在此笔者有以下几个建议供大家参考。

1）最好是朋友或熟人推荐，这样双方都比较放心。

2）开始之前先跟美工把报酬讲好，如果程序员和美工报酬一样的话，那就不要接。因为在项目后期中，程序会占绝大多数。

3）在洽谈过程中，一定要让客户把需求写成书面形式，然后再根据文本中要求的功能进行估价。这样更能明确我们的责任和义务，避免后期的纠纷。

4）和客户说清楚售后服务是收费还是免费，最好体现在合同中。

5）实现程序和页面的分离，这样不易发生什么误会，并能提高效率。

6）开工前先搞清楚客户说的报酬，是税前的还是税后的，因为很多单位到结账的时候会扣掉部分所谓的个人所得税。

7）最好有自己的服务器，把做的活放到自己的服务器上，如果客户满意了，付清全部的钱再把代码给他们，这样避免了客户不发钱，活已经给人家了，造成被动的局面。

第三篇　数 据 库 篇

第 13 章　数据库技术

数据库是“按照数据结构来组织、存储和管理数据的仓库”。在日常经济管理和日常工作中，常常需要把某些相关的数据放进这样的“仓库”，并根据管理需要进行相应的处理。在本章的内容中，将讲解和 Java Web 相关的数据库技术。通过本章能学到如下知识。

- 操作 MySQL。
- SQL Sever。
- 职场点拨——想寻找更好的工作。

2010 年 XX 月 XX 日，小雪

不知不觉间我已经工作了 1 年半，工资从当初的 3000 涨到了 4000。但我依旧是月光族，女友一直在催买房结婚，我感觉生活的压力好大。我很想寻找一份更好的工作……



一问一答

小菜：“我想寻找更好的工作，这里的待遇太低了！”

Wisdom：“你可得考虑好了，人往高处走这无可厚非，但是你能确保一定能找到更好待遇的工作吗？”

小菜：“这个……”

Wisdom：“嘿，你也不能确定吧，建议你看看本章最后的‘想跳槽了’，或许会给你带来一些启示！”

小菜：“嗯，好的。言归正传，本章的数据库有什么用？”

Wisdom：“你想通过跳槽来改变自己的命运，但是什么改变了软件行业的命运呢？答案是数据库，数据库是实现动态应用的最佳工具。数据库是当今大型应用程序开发必不可少的组成部分，通过数据库管理系统，可以有效地存储、管理和检索各种数据。”

13.1　认识 MySQL 数据库

在当前市面中有许多种数据库工具，下面将以 MySQL 为例作为本章的开篇。用 MySQL 建立一个数据库和一个表的过程十分简单，具体操作流程如下。

1）启动 MySQL 管理软件，单击“连接”按钮，打开“连接”窗口，在里面输入数据库的相关信息，单击“确定”按钮，如图 13-1 所示。



图 13-1 连接窗口

2）选择“localhost”，单击鼠标右键，在弹出的快捷菜单中选择“打开连接”命令，如图 13-2 所示。

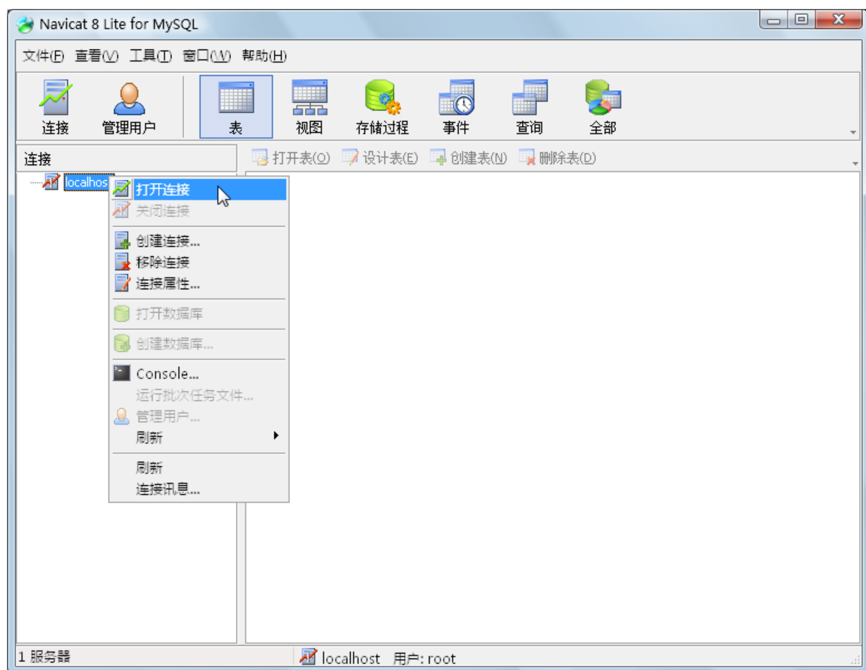


图 13-2 选择命令

3）打开“连接”后，仍然选择“localhost”选项，单击鼠标右键，在弹出的快捷菜单中选择“创建数据库”命令，打开“创建数据库”窗口，在数据库名文本框中输入“new”，然后单击“确定”按钮，如图 13–3 所示。

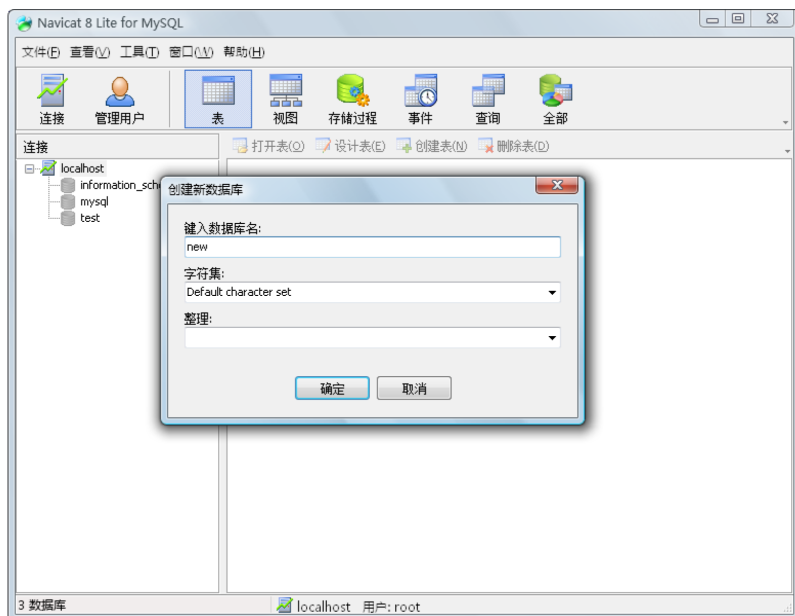


图 13–3 “创建数据库”窗口

4）建立好数据库后，双击建好的数据库，在打开的树形列表中选择“表”选项，然后单击鼠标右键，在弹出的快捷菜单中选择“创建表”命令，如图 13–4 所示。

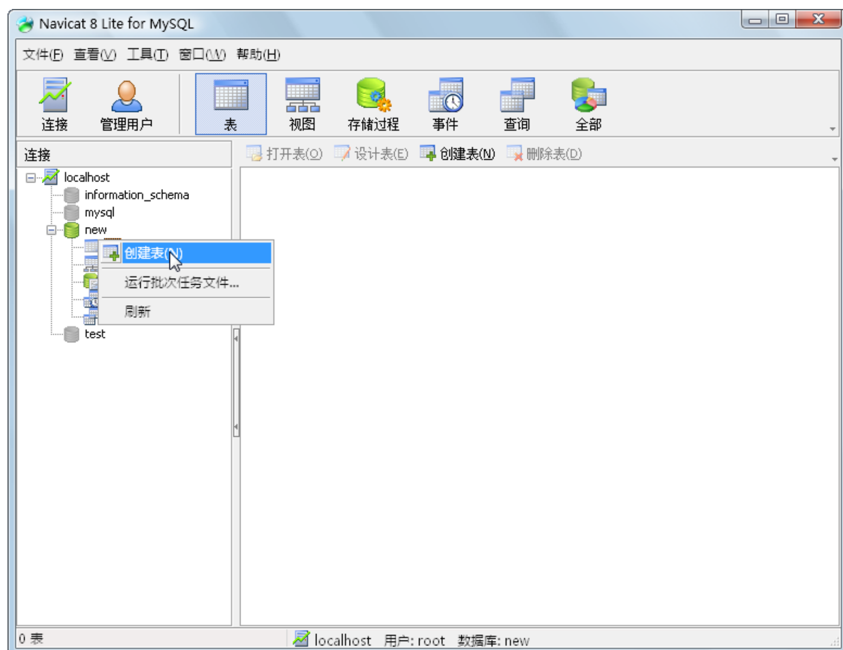


图 13–4 选择命令

5）选择命令后会打开“创建表”窗口，在此可以设置需要创建表的字段，如图 13-4 所示。

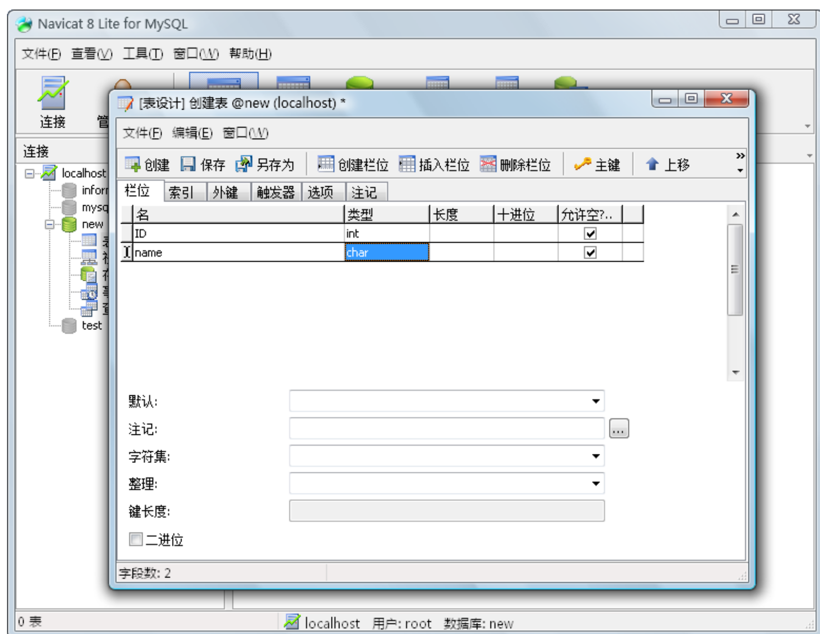


图 13-4　创建字段

6）创建字段后单击“保存”按钮，在打开的“表名”窗口中输入表名，输入完成后单击“确定”按钮，如图 13-5 所示。

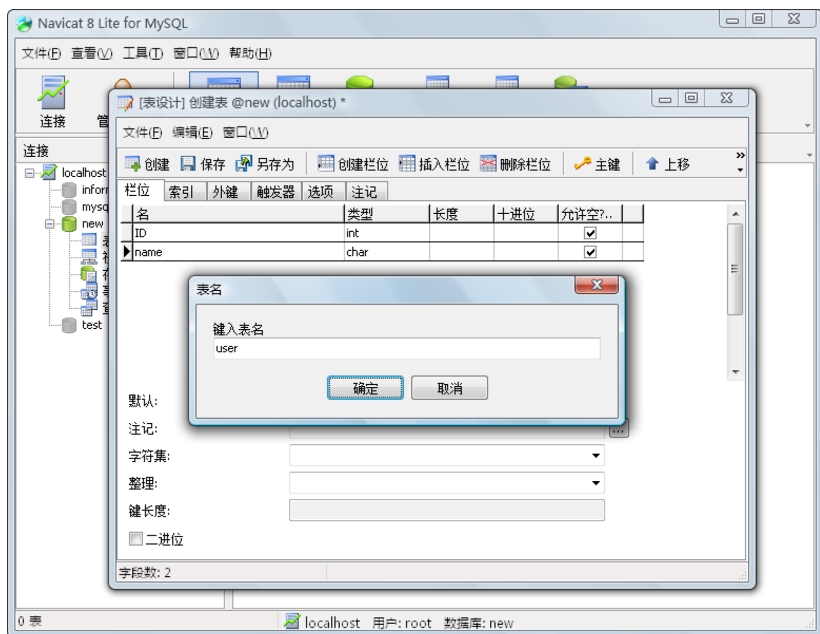


图 13-5　输入表名

7）在创建表之后可以输入记录，选择表并单击“打开表”按钮，然后输入记录，如图 13-6 所示。

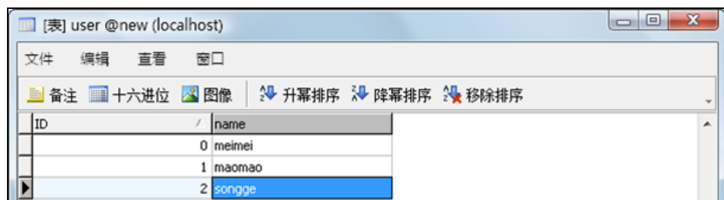


图 13-6　创建的记录

13.2　操作 MySQL

MySQL 是一个小型关系型数据库管理系统，广泛地应用在 Internet 上的中小型网站中。由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，被许多中小型网站所青睐。MySQL 的官方网站的网址是 www.mysql.com。

13.2.1　MySQL 的安装

读者可以去 MySQL 的官方网站或者其他资源网站获得 MySQL，由于是英语界面，对于初学者来说可能会有点麻烦。安装 MySQL 的具体操作流程如下。

1）双击下载的 MySQL 安装包后将会打开向导对话框，这是一个欢迎画面，在此单击“Next”按钮，如图 13-7 所示。

2）在打开的“Setup Type”对话框中选择一个选项，第一个选项是“典型安装”，第二个是“完全安装，第三个是“自定义安装”，这里选择第三个选项，然后单击“Next”按钮，如图 13-8 所示。

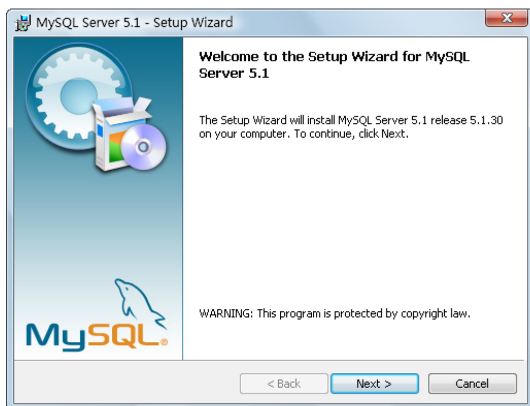


图 13-7　安装向导

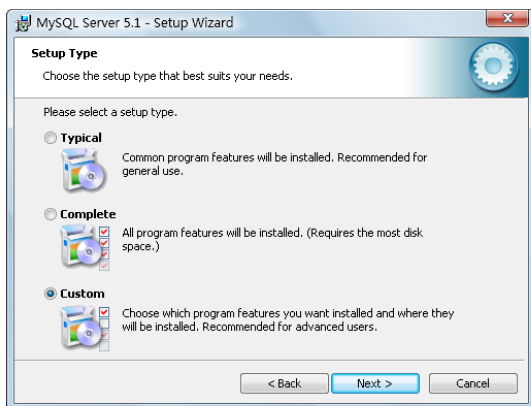


图 13-8　选择一个选项

3）打开“Custom Setup”对话框，在此可以设置安装的位置，这里保持默认设置，单击“Next”按钮，如图 13-9 所示。

4）在打开的对话框中继续单击“Next”按钮，等待一会将会打开一个对话框，单击“Next”按钮后又打开一个新对话框，接着单击“Next”按钮，如图 13-10 所示。

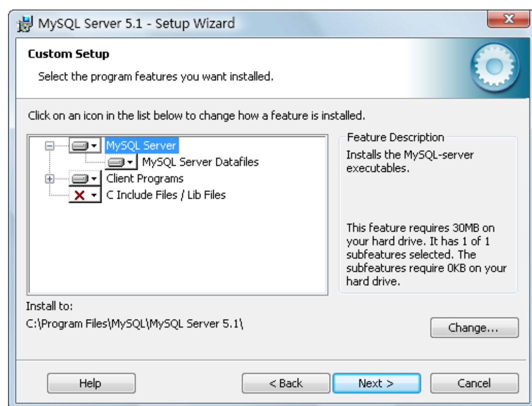


图 13-9　选择安装位置

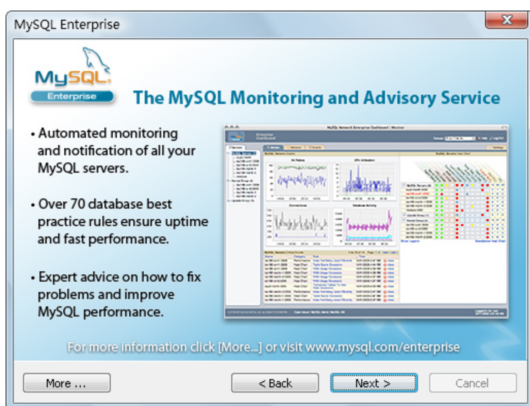


图 13-10　安装

5）单击“Next”按钮后将会打开如图 13-11 所示的对话框，此处只需要单击“Finish”按钮即可。

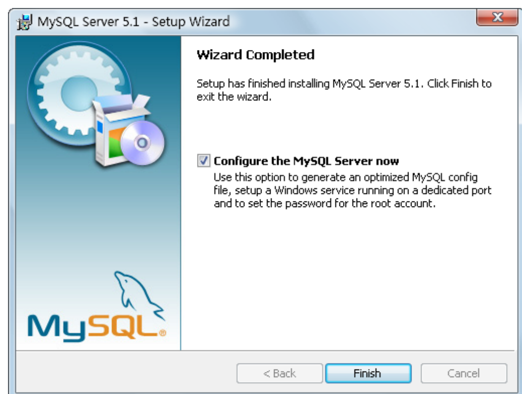


图 13-11　完成安装

13.2.2　设置 MySQL

第一次在计算机中安装 MySQL 后需要对 MySQL 进行专门设置。在图 13-11 中只需要单击“Finish”按钮就会打开一个新的对话框，此时开始对 MySQL 进行设置。设置 MySQL 数据库的步骤如下。

1）打开如图 13-12 所示的对话框，然后单击“Next”按钮。

2）在打开的对话框中有两个选项，第一个是详细设置，第二个是标准设置，在此选择第一个选项，单击“Next”按钮，如图 13-13 所示。

3）在打开的对话框中有 3 个选项，第一个是开发者，第二个是服务器，第三个是专业版，在此选择第三个选项，如图 13-14 所示。

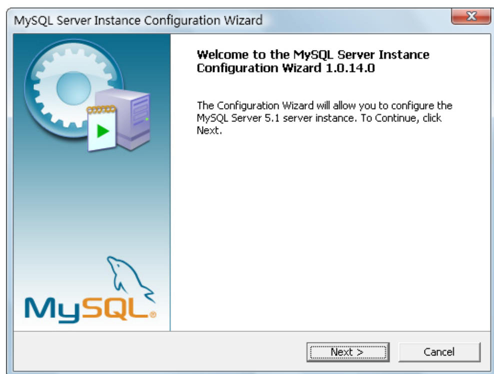


图 13-12　欢迎界面

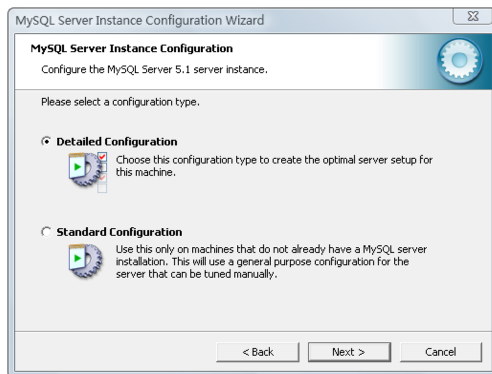


图 13-13　完全设置

4）在新打开的对话框中有 3 个选项，在此选择第一个选项，如图 13-15 所示。

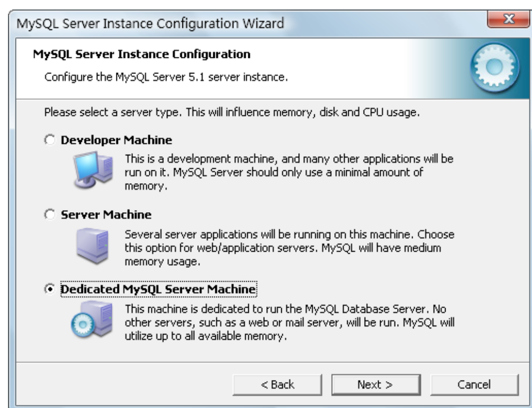


图 13-14　选择需要的服务

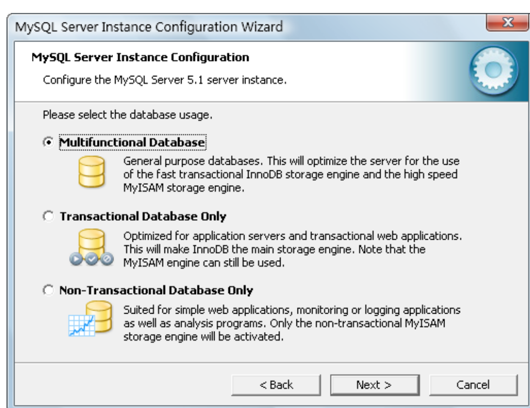


图 13-15　选择一个选项

5）在打开的对话框中设置日志的位置，在此保持默认设置，如图 13-16 所示。

6）在新打开的对话框中有新的选项，在此选择第三个选项，然后在下拉列表中选择一个数，读者可以根据自己的需要进行手动输入，如图 13-17 所示。

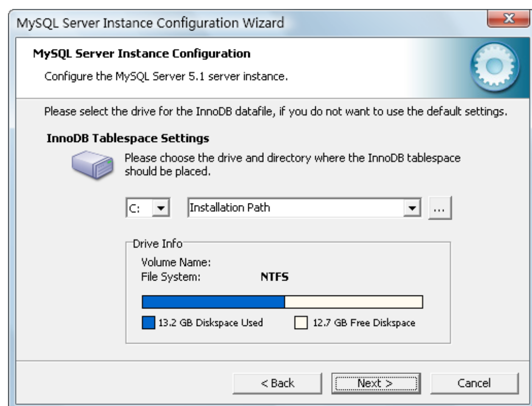


图 13-16　设置日志位置

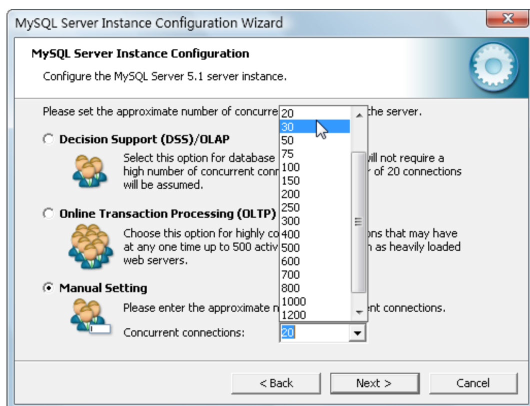


图 13-17　手动设置

7）在新打开的对话框中设置服务器端口，在此可以保持默认设置，单击“Next”按钮，如图 13-18 所示。

8）在打开的对话框中选择第三个选项，在下拉列表中选择“gb2312”选项，然后单击“Next”按钮，如图 13-19 所示。

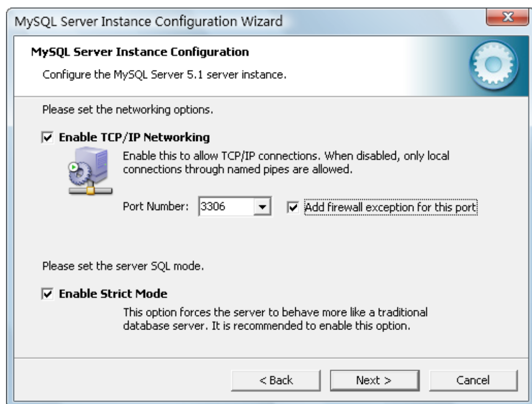


图 13-18　设置端口号

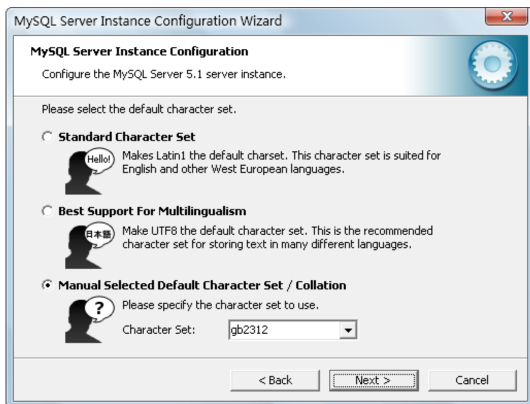


图 13-19　设置语言选项

9）在打开的对话框中单击“Next”按钮，然后在新打开的对话框中设置密码，设置好后勾选所有的复选框，如图 13-20 所示。

10）在打开的对话框中单击“Execute”按钮，当执行命令完成后，单击“Finish”按钮，如图 13-21 所示。



图 13-20　设置密码

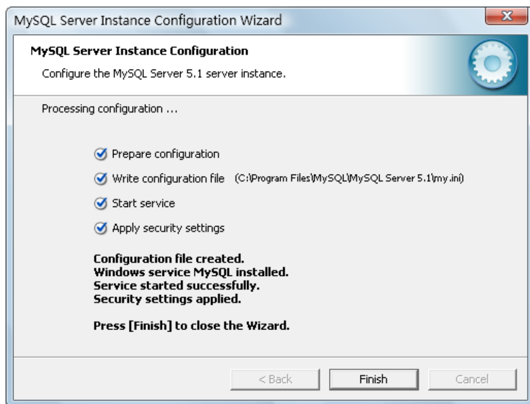


图 13-21　执行命令

13.2.3　登录 MySQL

安装 MySQL 完成后，就可以登录 MySQL 数据库了。登录 MySQL 的方法十分简单，只需要在程序中选择“MySQL Command Line Client”即可。登录 MySQL 的具体操作步骤如下。

1）在打开的窗口中输入登录密码，按〈Enter〉键进入数据库系统，如图 13-22 所示。

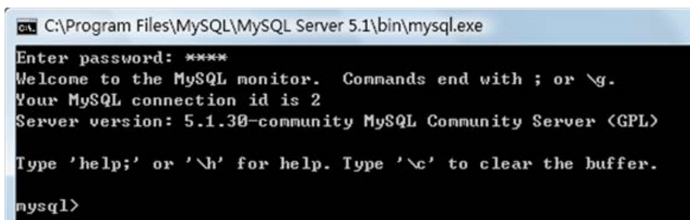


图 13-22　登录数据库系统

2）进入数据库后可以使用各种 SQL 命令操作数据库，例如查询数据库系统有哪些数据库，则可以输入 show databases 命令，然后按〈Enter〉键，如图 13-23 所示。

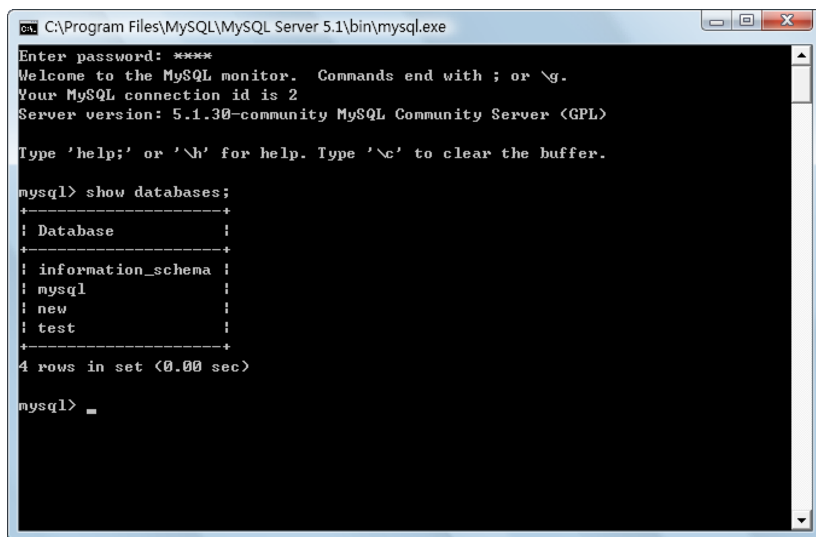


图 13-23　对数据库进行操作

上述操作数据库的方法很不方便，读者可以通过各种管理软件实现对 MySQL 的管理，如 SQLyog、Navicat Lite for MySQL 和 MySQL-Front，在本书中使用了 Navicat Lite for MySQL。

13.2.4　创建 MySQL 数据库

安装完 MySQL 数据库和管理软件后，就可以随时快速创建需要的数据库了，创建数据库的方法十分简单，具体流程如下。

1）启动 Navicat Lite for MySQL 管理软件，在窗口中选择“localhost”选项，单击鼠标右键，在弹出的快捷菜单中选择“新建数据库”命令，如图 13-24 所示。

2）打开“创建数据库”对话框，在“键入数据库名称”文本框中输入“news”，然后单击“确定”按钮，如图 13-25 所示。

13.2.5　创建 MySQL 数据库表

创建好数据库后，就可以创建表了，创建过程十分简单。不过在创建表的时候，一定要

明白什么是字段，什么是字段的类型。其实字段与变量是对应的，类型与 Java 中的数据类型也是对应的，下面将讲解如何创建表。

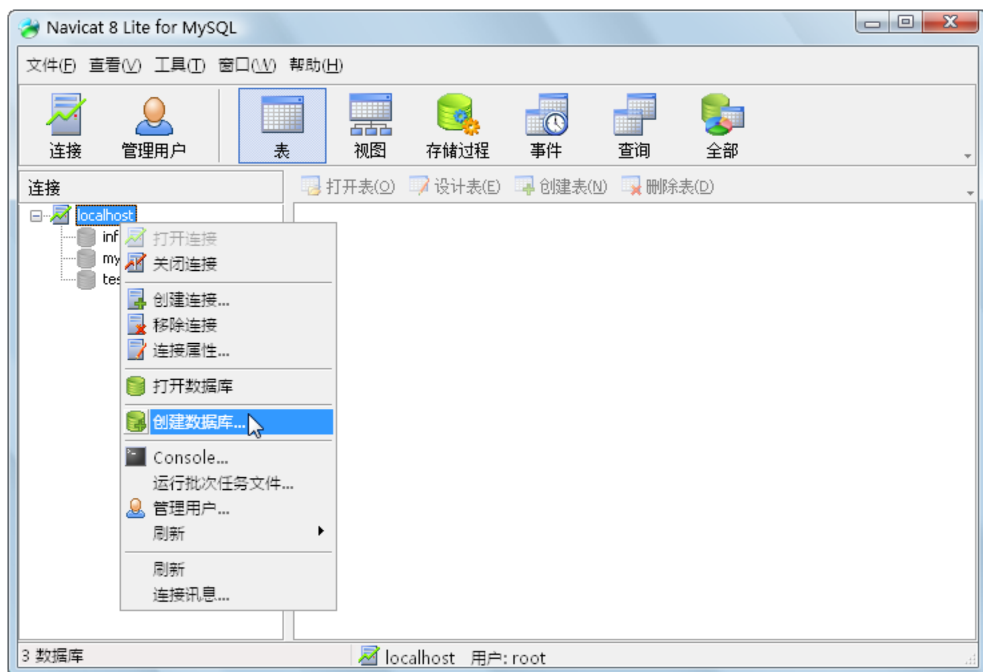


图 13-24　选择命令

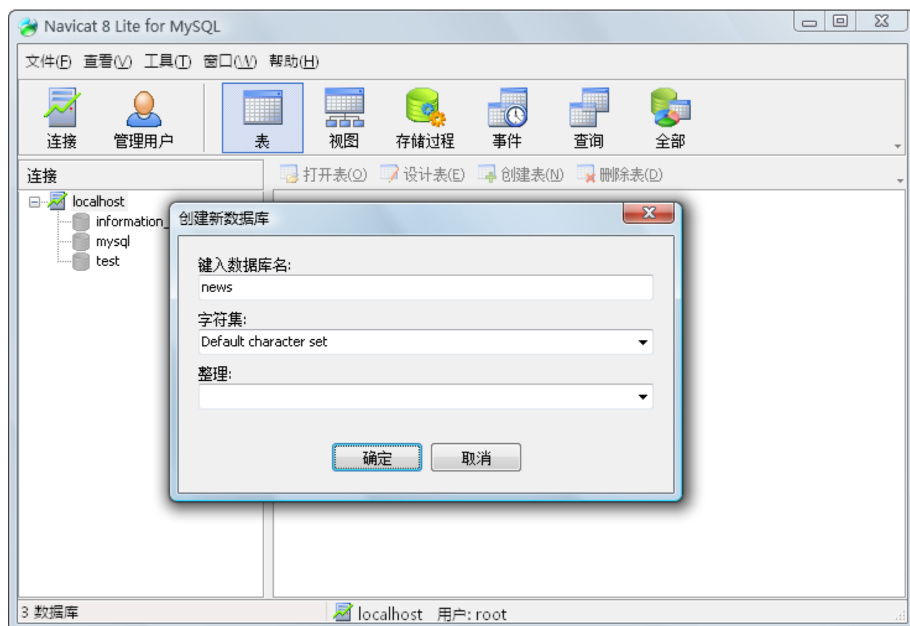


图 13-25　“创建新数据库”对话框

1）双击需要创建表的数据库，在展开的树形列表中选择“表”选项，单击鼠标右键，在弹出的菜单中选择“创建表”命令，如图 13-26 所示。



图 13-26 选择命令

2）打开“表设计”窗口，在对应位置设置字段属性，设置后如果需要继续创建字段，可以单击“创建栏位”继续创建字段，如图 13-27 所示。

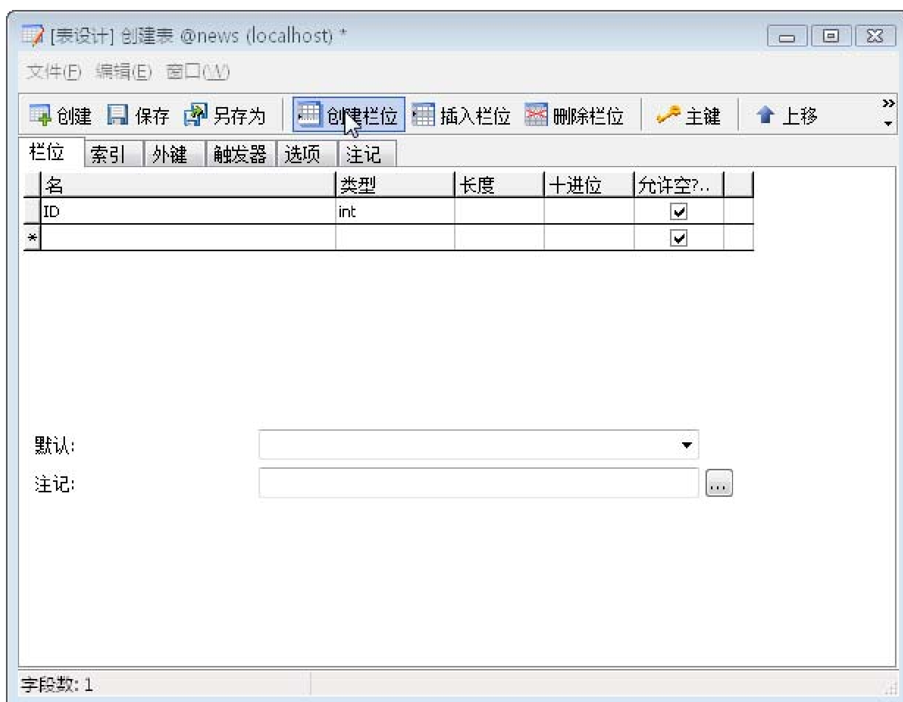


图 13-27 设计字段

3）设置字段后单击“保存”按钮，打开“表名”对话框，在键入表名文本框中输入“book”，然后单击“确定”按钮，如图 13-28 所示。

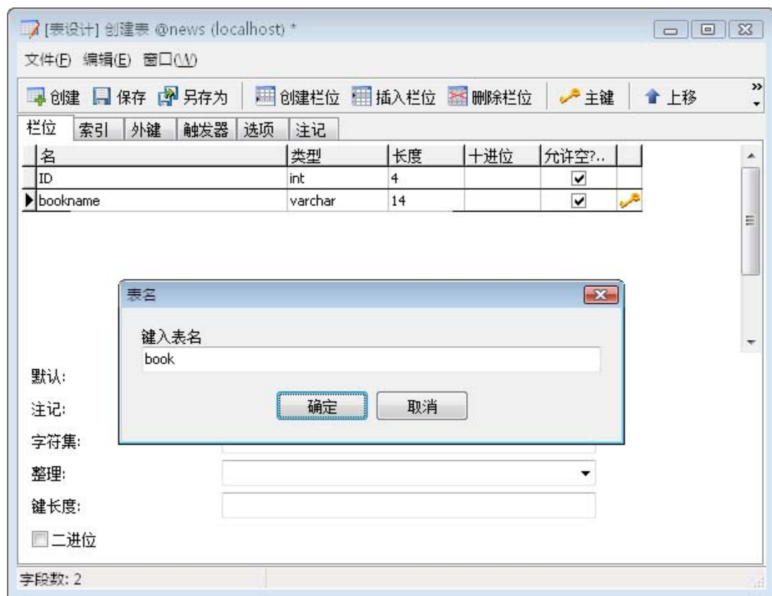


图 13-28　输入表名

13.2.6　输入记录

记录就是信息，是对数据库表中某一行的称呼。记录对于数据库来说十分重要，下面讲解在数据库中输入记录的操作流程。

1）选择需要创建记录的表并打开，然后单击“打开表”按钮，如图 13-29 所示。

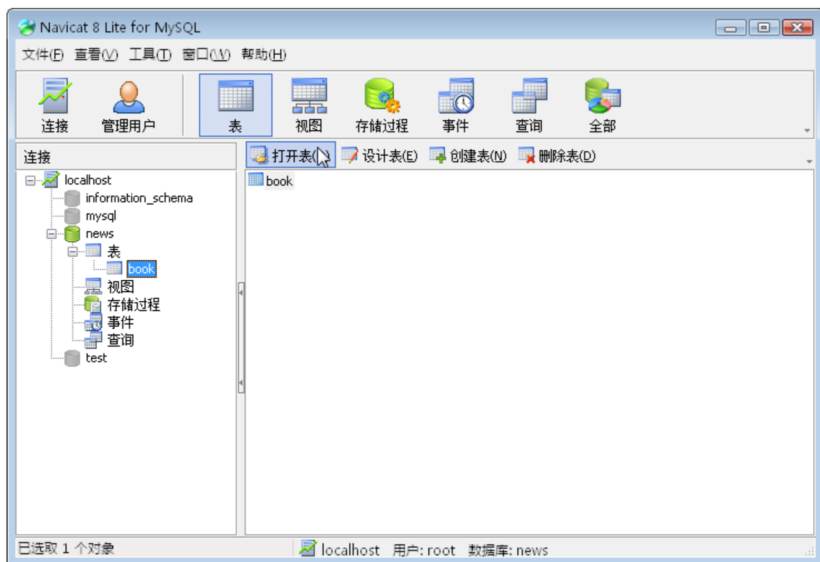


图 13-29　打开表

2）在设置字段输入对应的值，如果需要插入新的一条记录，单击左下角的“+”按钮，如图 13-30 所示。

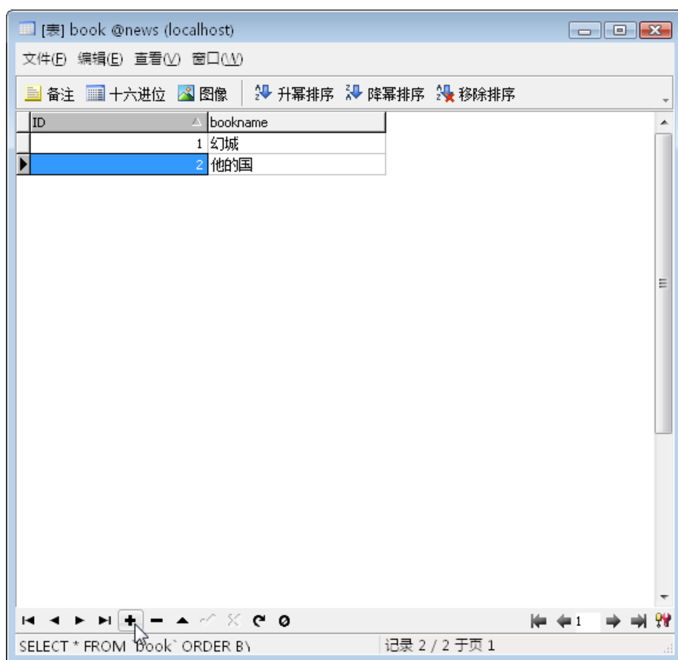


图 13-30　添加记录

3）当完成记录的创建后，可以直接关闭右上角的关闭按钮，如图 13-31 所示。

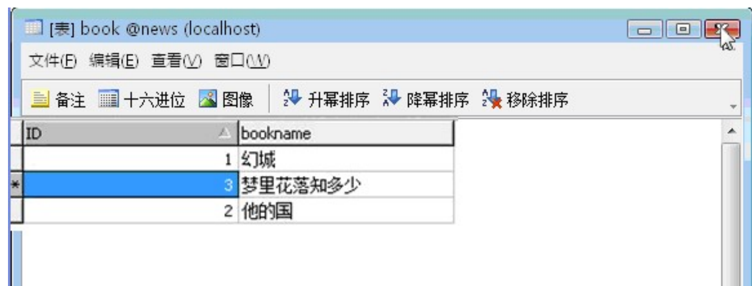


图 13-31　关闭窗口

13.3　SQL Sever 可以这么简单

SQL Sever 数据库是 Microsoft 公司的优秀产品，是当前市面中的主流数据库工具。在本节的内容中，将详细讲解使用 SQL Sever 2000 的流程。

13.3.1　创建 SQL Sever 数据库

1）当安装 SQL Sever 2000 后，单击开始菜单，选择“所有程序/Microsoft SQL Sever/企业管理器”命令，如图 13-32 所示。

2）启动企业管理器后，用户需要在左边的树形列表中选择“数据库”选项，然后单击鼠标右键，选择“新建数据库”命令，如图 13-33 所示。

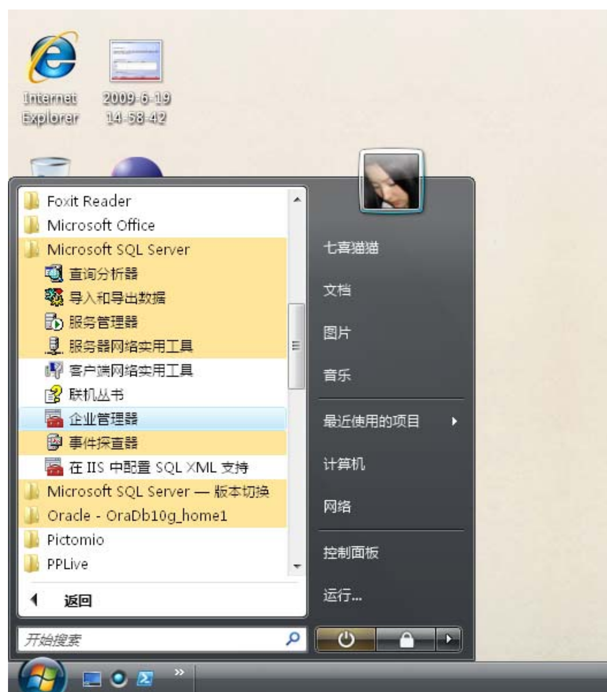


图 13-32　选择命令启动 SQL Sever 2000

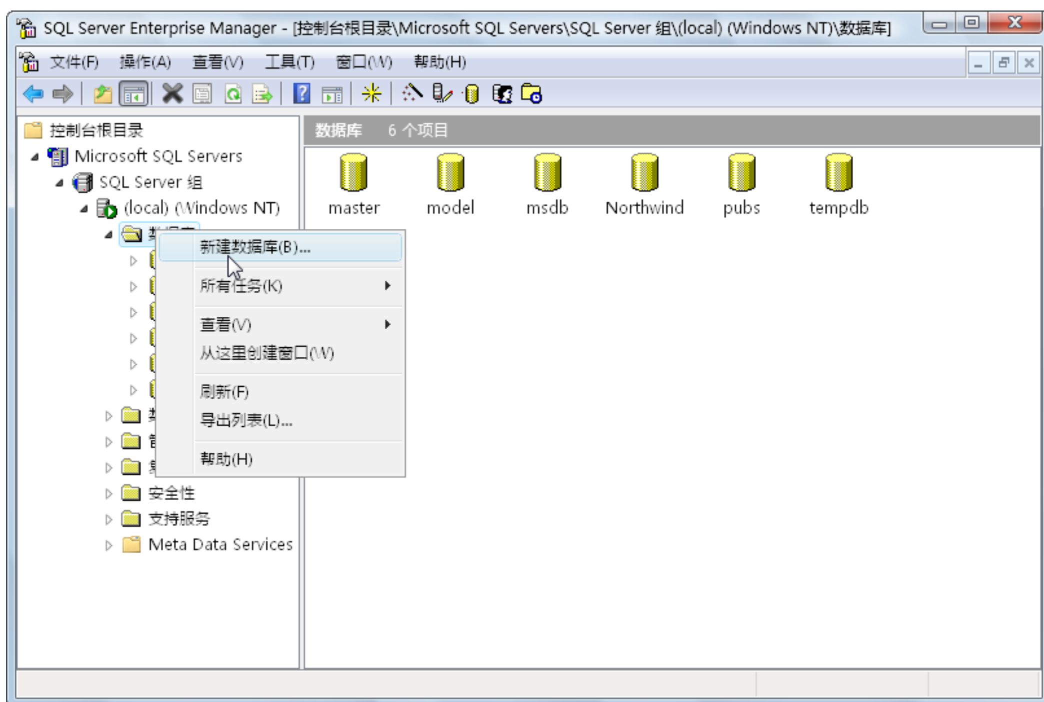


图 13-33　新建数据库

3）打开“数据库属性”对话框，在“名称”文本框中输入“news”，如图 13-34 所示。

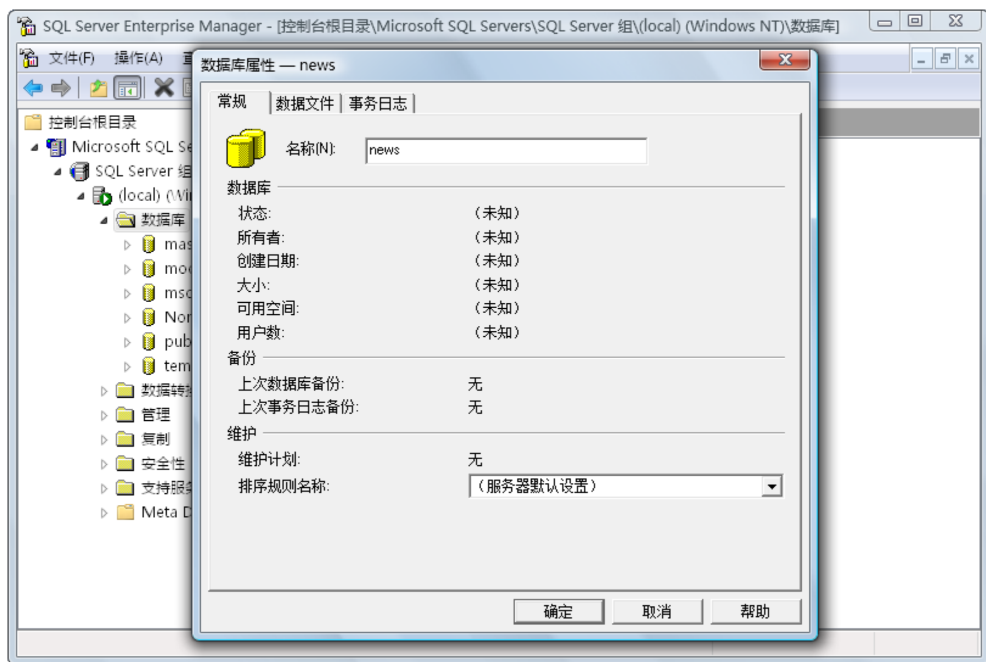


图 13-34　输入数据库名称

4）单击“数据文件”选项卡设置数据库文件的位置，在默认情况下将数据库放在默认目录下。可以单击按钮打开“查找数据库文件”，此时可以根据需要指定一个位置，如 F 盘，如图 13-35 所示。

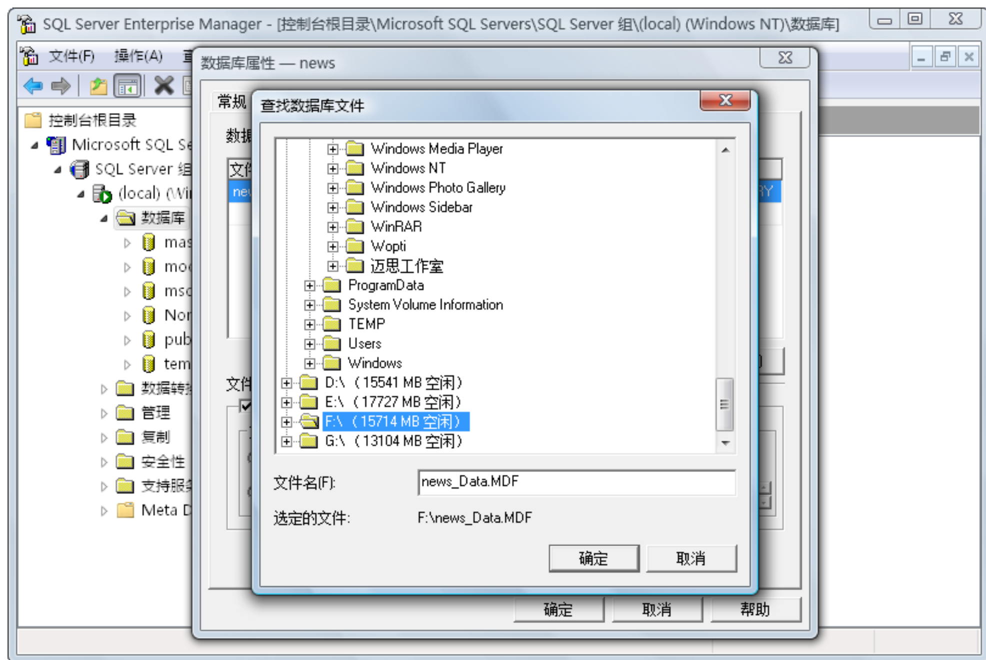


图 13-35　更改数据库文件

5）单击“事务日志”选项卡，设置数据库文件的位置。在默认情况下，是将数据库放在系统盘下的。可以单击 按钮打开“查找事务日志文件”，此时可以根据需要指定一个位置，如 F 盘，如图 13-36 所示。

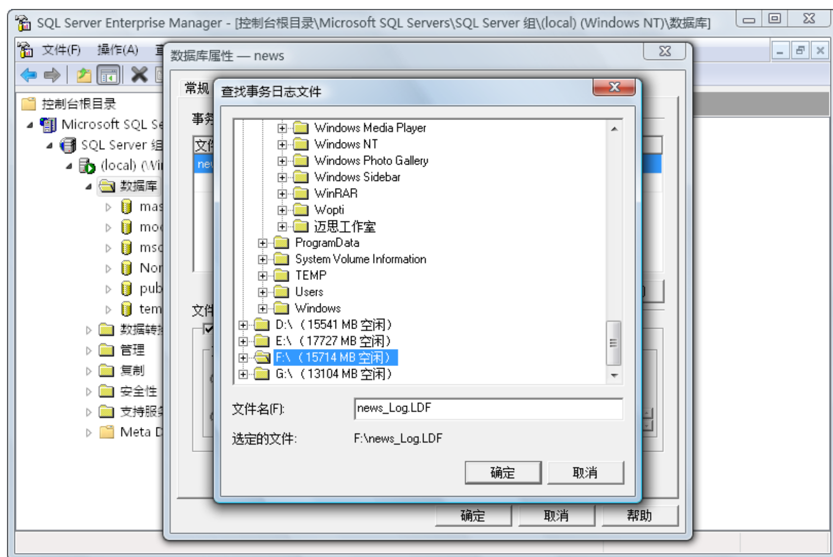


图 13-36　指定事务日志文件的位置

6）设置完成后，依次单击“确定”按钮完成创建 SQL Sever 数据库工作。

13.3.2　创建表

当创建好数据库后，用户可以创建数据库中的表，其具体操作步骤如下。

1）单击需要创建表的数据库，在属性列表中选择“表”选项，单击鼠标右键，选择“新建表”命令，如图 13-37 所示。

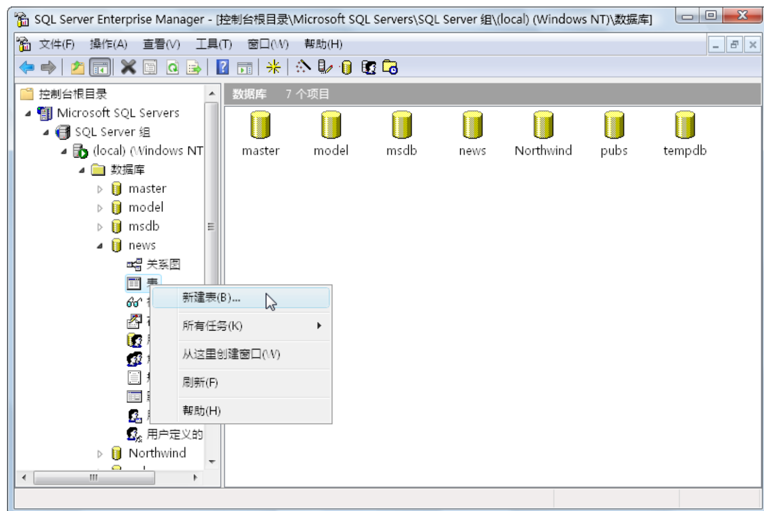


图 13-37　选择“新建表”命令

2）此时会打开一个新窗口，接下来需要为这个表创建字段，然后为字段设置属性，如图 13-38 所示。

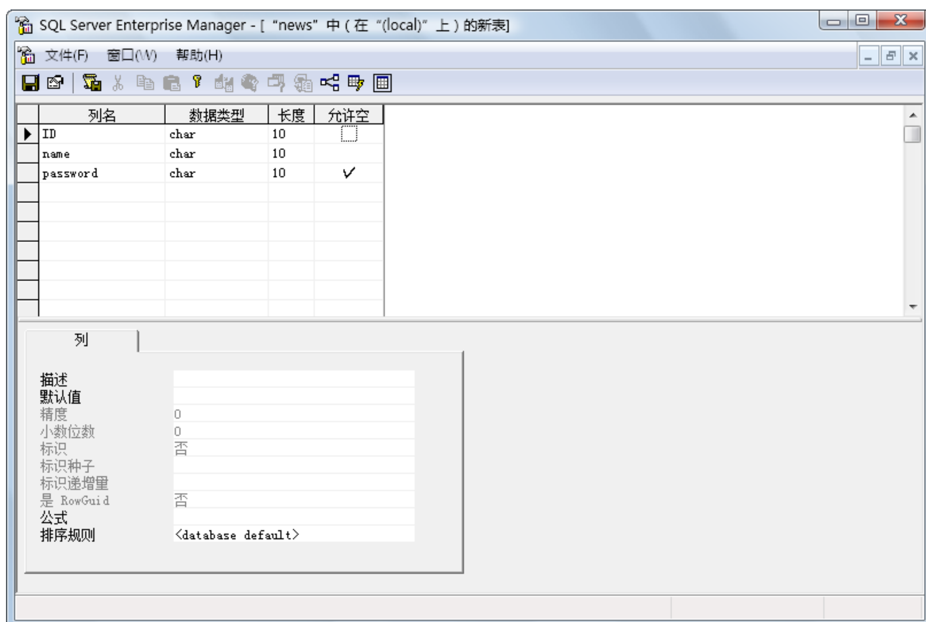


图 13-38　设置字段

3）设置完成后，单击“保存”按钮后打开“选择名称”对话框，在“输入表名”文本框中输入名称，如“user”，单击“确定”按钮，如图 13-39 所示。

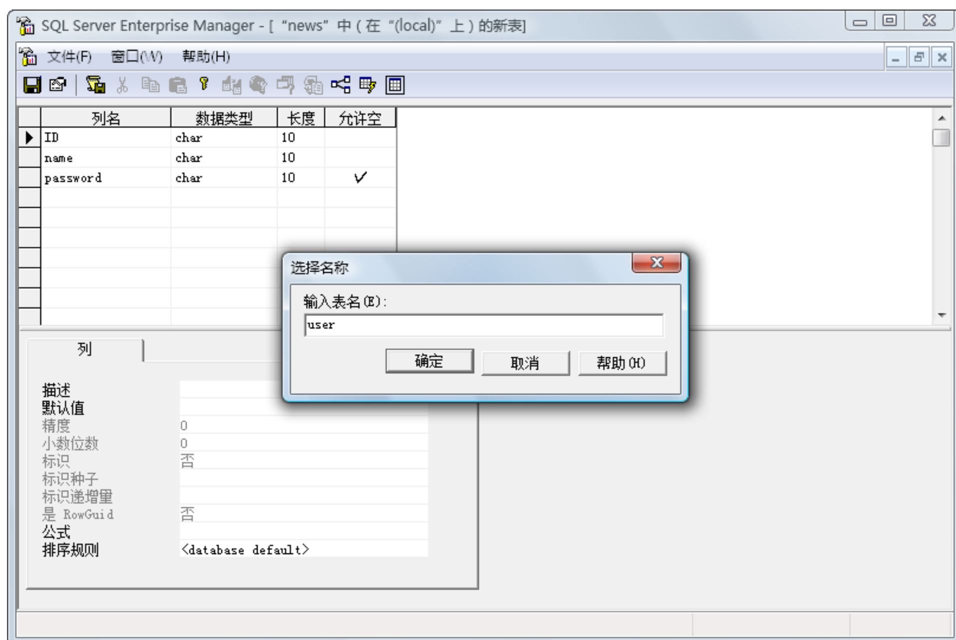


图 13-39　输入表名

4）关闭最里面的窗口，可以返回数据库中的窗口来查看刚刚新建的表，如图 13-40 所示。

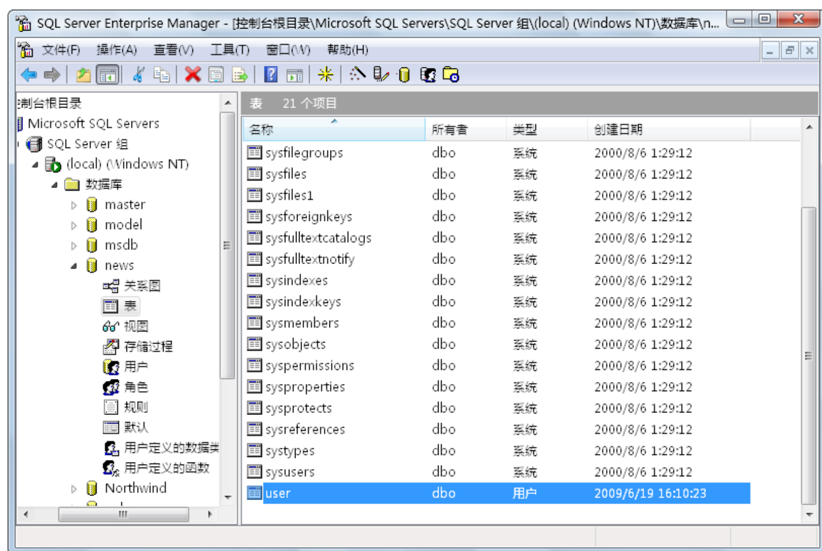


图 13-40 新建的表

13.3.3 创建记录

当创建一个表后，接下来需要在表中输入记录。输入记录的过程十分简单，与 MySQL 和 Access 数据库的操作方法十分类似，具体操作流程如下。

1）选择需要创建记录的表，如“user”表，然后单击鼠标右键，在弹出的快捷菜单中选择“打开表/返回所有行”命令，如图 13-41 所示。

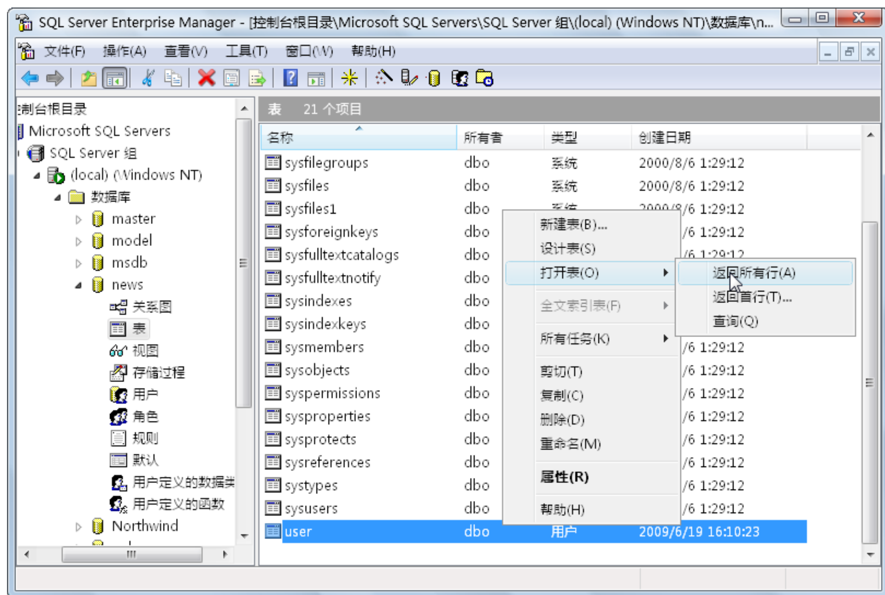


图 13-41 返回所有行命令

2）输入完成后可以将它直接关闭，相应的记录也将被输入，如图 13-42 所示。



图 13-42　输入记录



在输入记录时，输入的数据一定要和创建表时设置的字段相同，否则在输入记录时会报错，如图 19-43 所示。输入完成后，一定要单击内部的关闭按钮，如果关闭外部的按钮是表示退出软件。这是初学者容易犯的错误。

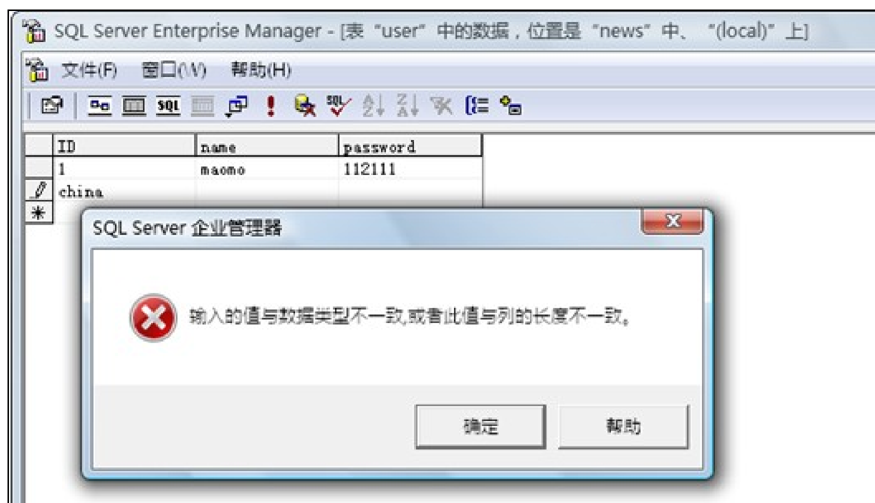


图 19-43　输入的数据类型不符合

13.4　疑难问题解析

在本章的内容中，详细讲解了数据库技术的基本知识。本节中，将对本章中比较难理解的问题进行讲解。

读者疑问：除了 MySQL、SQL Server 这些数据库软件，还有其他数据库软件吗？使用时，该选择那一款数据库软件呢？

解答：数据库软件很多，除了本章讲的数据库软件外，还有 DB2、Oracle、Access 等，它们都是有名的数据库，用户可以根据自己的需要选择数据库软件。MySQL 是免费开源的，在中小型企业开发可以考虑使用它，大型企业用户不妨考虑使用 SQL Sever 或者 Oracle 数据库，它们使用的方法都是大同小异，没有大的差别。

读者疑问：在讲解 MySQL 时曾经提起过 SQL 语句，说用 SQL 语句可以方便地操作数据库系统，是吗？

解答：SQL 语句是数据库的一个标准，只要在权限范围内，它是可以完全操作数据库的，在编程时这些语句尤为重要，下一章将会为大家讲解 SQL 语句，总的来说，它可以分为 4 种，介绍如下。

- 数据查询语句（SELECT 语句）。
- 数据操纵语句（INSERT, UPDATE, DELETE 语句）。
- 数据定义语句（如 CREATE, DROP 等语句）。
- 数据控制语句（如 COMMIT, ROLLBACK 等语句）。



职场点拨——想寻找更好的工作

人往高处走，水往低处流，在职场中的程序员们为了获取更好的待遇，肯定会有寻找更好工作的想法。本文将简单谈谈寻找更好工作的一些经验。

（1）让旧雇主和自己皆大欢喜

离职是发生劳动争议的导火索，据劳动法网的调查，近几年最受关注的劳动关系问题中，解除合同以 26.7%的高比率占据了头把交椅。可见如果处理不好离职问题，很容易造成劳动纠纷。很多程序员存在一个误区，总以为反正是快要走了，趁此机会把压抑在心底的所有不满都发泄出来，结果将公司的气氛弄得乌烟瘴气，最后落得个不欢而散。在此建议在离职之前，妥善地处理好这些事情，使自己和上司皆大欢喜，这会为今后的职场形象添加一抹重彩。

（2）找一个委婉的理由

当告诉上司你的决定的时候，必须要说得委婉些。最忌讳说“由于单位现在状况不太好”之类的话，而应该找一些上司听起来更顺耳的话来解释，尽量从家庭方面找原因，而不要从当前单位找原因。如果尖锐地一语点破，会挫伤领导的自尊心。

（3）一定做好交接工作

离开公司时，需要将一些工作交接给下任。在交接时一定要仔细，不留任何纰漏。并且要严格地设置一个时间，在交接期间不能有错误的想法，例如，认为反正快要走了，不妨把压抑在心里的不满发泄出来。世界其实很小，树敌不是聪明的选择。

（4）避谈新单位

不要在上司或者同事面前说新单位的待遇是如何好，这种不自觉的优越感会给别人带来不舒服的感觉。聪明的做法是对领导和同事表示感谢。

第 14 章　使用 JDBC 连接数据库

Java 开发离不开数据库。要想操作数据库，前提是必须建立和数据库的连接。Java 和其他语言不一样，必须使用 JDBC 连接数据库。JDBC（Java Data Base Connectivity，Java 数据库连接）是一种用于执行 SQL 语句的 Java API，可以为多种关系数据库提供统一访问，它由一组用 Java 语言编写的类和接口组成。JDBC 为工具/数据库开发人员提供了一个标准的 API，以此可以构建更高级的工具和接口，使数据库开发人员能够用纯 Java API 编写数据库应用程序。通过本章能学到如下知识。

- 什么是 JDBC。
- 连接 MySQL 数据库。
- 连接 SQL Sever 2000。
- 连接其他数据库的方法。
- 职场点拨——准备创业。

2010 年 XX 月 XX 日，小雪

我最近压力依旧特别大，很多地方都需要花钱。今天听大学同学 A 说：同学 B 创业有成，买房、买车，并办了一个风光的婚礼……听后我很羡慕。我也想通过创业来改变自己的命运……


一问一答

小菜：“我想个人创业，不知是否可行？”

Wisdom：“很不错，原来想跳槽，现在想创业，都是积极向上的想法。创业成功会使你得到想要的一些东西，但是万一失败，你也会失去很多东西，并对你的信心带来很大的冲击。所以你得三思而后行，建议你参考本章最后的‘准备创业’。”

小菜：“嗯，我会好好考虑的！言归正传，本章的 JDBC 重要吗？”

Wisdom：“JDBC 是操作数据库数据的一种方法，通过它可以很好地控制数据库内的数据。”

14.1　连接数据库

MySQL 是常用的数据库工具，下面以一段简单的代码为本章的开篇，让读者理解连接

数据库的重要性和关键技术。具体代码如下所示。

```
        import java.sql.*;
        public class JDBCTest
        {
        public static void main(String[] args)
        {
                // 驱动程序名
                String driver = "com.mysql.jdbc.Driver";
                // URL 指向要访问的数据库名 scutcs
                String url = "jdbc:mysql://127.0.0.1:3306/scutcs";
                // MySQL 配置时的用户名
                String user = "root";
                 // MySQL 配置时的密码
                String password = "1234";
                try {
                 // 加载驱动程序
                 Class.forName(driver);
                 // 连续数据库
                 Connection  conn  =  DriverManager.getConnection(url,  user,
password);
                 if(!conn.isClosed())
                  System.out.println("Succeeded connecting to the Database!");
                 // statement 用来执行 SQL 语句
                 Statement statement = conn.createStatement();
                 // 要执行的 SQL 语句
                 String sql = "select * from student";
                 // 结果集
                 ResultSet rs = statement.executeQuery(sql);
                 System.out.println("-----------------");
                 System.out.println("执行结果如下所示:");
                 System.out.println("-----------------");
                 System.out.println(" 学号" + "\t" + " 姓名");
                 System.out.println("-----------------");
                 String name = null;
                 while(rs.next()) {
                   // 选择 sname 这列数据
                  name = rs.getString("sname");
                     // 首先使用 ISO-8859-1 字符集将 name 解码为字节序列并将结果存储新
                    // 的字节数组中。
                  // 然后使用 GB2312 字符集解码指定的字节数组
                  name = new String(name.getBytes("ISO-8859-1"),"GB2312");
                  // 输出结果
                  System.out.println(rs.getString("sno") + "\t" + name);
                 }
                 rs.close();
                 conn.close();
```

```
            } catch(ClassNotFoundException e)
    {
             System.out.println("Sorry,can`t find the Driver!");
             e.printStackTrace();
            } catch(SQLException e)
   {
             e.printStackTrace();
            } catch(Exception e) {
             e.printStackTrace();
            }
   }
   }
```

上述代码执行后会得到如图 14-1 所示的效果。

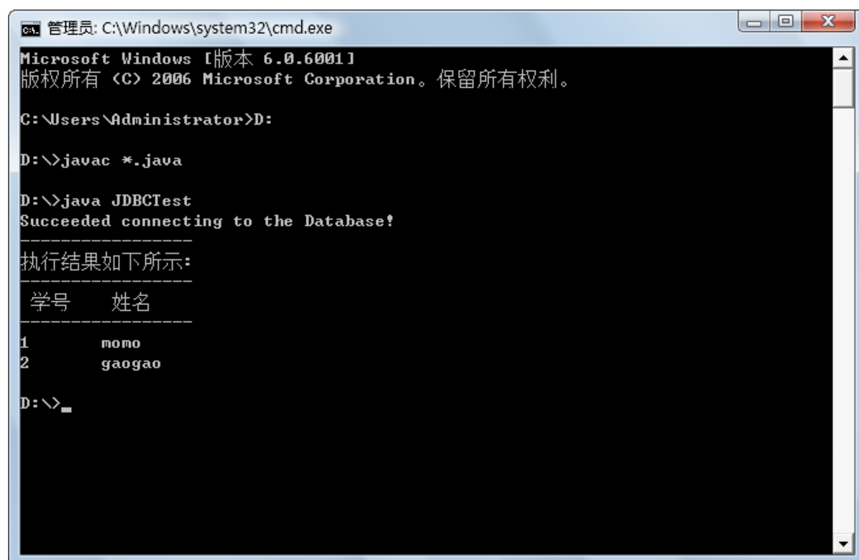


图 14-1　执行 MySQL 查询的结果

14.2　什么是 JDBC

有了 JDBC 后，就可以向各种关系数据发送 SQL 语句，就不必为访问 Sybase 数据库专门写一个程序，再为访问 Oracle 数据库又专门写一个程序，或为访问 Informix 数据库又编写另一个程序等。程序员只需用 JDBC API 写一个访问程序就足够了。同时，将 Java 语言和 JDBC 结合后，程序员不必为不同的平台编写不同的应用程序，只需写一个程序即可在任何平台上运行，这正是 Java 语言跨平台的优势。

14.2.1　JDBC API

JDBC 能直接调用 SQL 命令，这是一个“低级”接口，比其他数据库连接的 API 易于使

用。同时 JDBC 也被设计为一种基础接口，在它之上可以建立更加高级的接口和工具。高级接口是“对用户友好的”接口，使用的是一种更易理解和更为方便的 API，这种 API 在幕后被转换为诸如 JDBC 这样的低级接口。

在关系数据库的“对象/关系”映射中，表中的每行对应类的一个实例，而每列的值对应该实例的一个属性。所以程序员可直接对 Java 对象进行操作，存取数据所需的 SQL 调用将在后台自动生成。也可以提供更复杂的映射，例如，将多个表中的行结合进一个 Java 类中。

随着开发人员对 JDBC 的了解，越来越多的开发人员使用基于 JDBC 的工具。程序员一直在编写使最终用户对数据库的访问变得更为简单的应用程序。例如，编写一个应用程序，这个程序能够提供一个选择数据库任务的菜单。当任务被选定后，应用程序将给出提示及空白供填写执行选定任务所需的信息。所需信息输入应用程序将自动调用所需的 SQL 命令。在这样一种程序的协助下，即使用户根本不懂 SQL 的语法，也可以执行数据库任务。

14.2.2　JDBC 对 B/S 和 C/S 模式的支持

JDBC API 既支持数据库访问的两层模型（C/S），同时也支持三层模型（B/S）。在两层模型中，Java applet 或应用程序将直接与数据库进行对话。这需要一个 JDBC 驱动程序来与所访问的特定数据库管理系统进行通信。用户的 SQL 语句被送往数据库中，而结果将被送回给用户。在这种模式下，数据库可以存放在另一台计算机上，用户可以通过网络连接到数据库。这种模式叫做客户机/服务器配置，其中用户的计算机为客户机，提供数据库的计算机为服务器。既可以是 Intranet 网络（它可将公司职员连接起来），也可以是 Internet 网络。

在三层模型中，命令先是被发送到服务的“中间层”，然后在中间层将 SQL 语句发送给数据库。数据库对 SQL 语句进行处理并将结果送回中间层，中间层再将结果送回用户。使用中间层的好处是，用户可以利用易于使用的高级 API，而中间层将把它转换为相应的低级调用。在许多情况下，三层结构可提供一些性能上的好处。

原来中间层通常是用 C 或 C++语言来编写的，这些语言执行速度较快。近几年随着最优化编译器（它把 Java 字节代码转换为高效的特定于机器的代码）的引入，用 Java 来实现中间层将变为可能。这样可以充分利用 Java 的诸多优点来实现中间层。

14.2.3　JDBC 驱动类型

JDBC 是应用程序编程接口，描述了一套访问关系数据库的标准 Java 类库，并且还为数据库厂商提供了一个标准的体系结构，让厂商可以为自己的数据库产品提供 JDBC 驱动程序。这些驱动程序可以让 Java 应用程序直接访问厂商的数据产品，从而提高了 Java 程序访问数据库的效率。在 Java 程序设计中，JDBC 驱动可以分为以下 4 种。

- JDBC-ODBC 桥：是微软公司开放服务结构（Windows Open Services Architecture，WOSA）中有关数据库的一个组成部分，它建立了一组规范，并提供了一组对数据库访问的标准 API（应用程序编程接口）。这些 API 利用 SQL 来完成其大部分任务，并且 ODBC 本身也提供了对 SQL 语言的支持，用户可以直接将 SQL 语句送给

ODBC，因为大部分数据库都支持通过 ODBC 来访问。

- ❑ 本地 API 驱动：本地 API 驱动直接把 JDBC 调用转变为数据库的标准调用再去访问数据库，这种方法需要本地数据库的驱动代码。
- ❑ 网络协议驱动：这种驱动是根据三层结构建立的，JDBC 先把对数据库的访问请求传递给网络上的中间件服务器，中间件服务器再把请求翻译为符合数据库规范的调用，把这种调用传给数据库服务器。如果中间件服务器也是用 Java 开发的，那么在中间层也可以使用上面两种类型的 JDBC 驱动程序。
- ❑ 本地协议驱动：这种驱动直接把 JDBC 调用转换为符合相关数据库系统规范的请求。由于驱动的应用直接和数据库服务器通信，Java 程序可以完全实现，因此实现了平台独立性，本地协议驱动—数据库 Server 由于这种驱动不需要先把 JDBC 的调用传给 ODBC 或本地数据库接口或者是中间层服务器，所以它的执行效率是非常高的。而且，它根本不需要在客户端或服务器端装载任何软件或驱动。这种驱动程序可以动态地被下载，但是对于不同的数据库需要下载不同的驱动程序。

14.3 轻松连接 MySQL 数据库

当前大多数中小型企业都选择 MySQL 作为数据库的解决方案。

14.3.1 下载 MySQL 的 JDBC 驱动

MySQL 的 JDBC 驱动是很好下载的，用户可以通过搜索引擎搜索资源并下载。如果读者对英文比较熟悉，也可以去 MySQL 官方网站下载 JDBC 驱动，如图 14-2 所示。

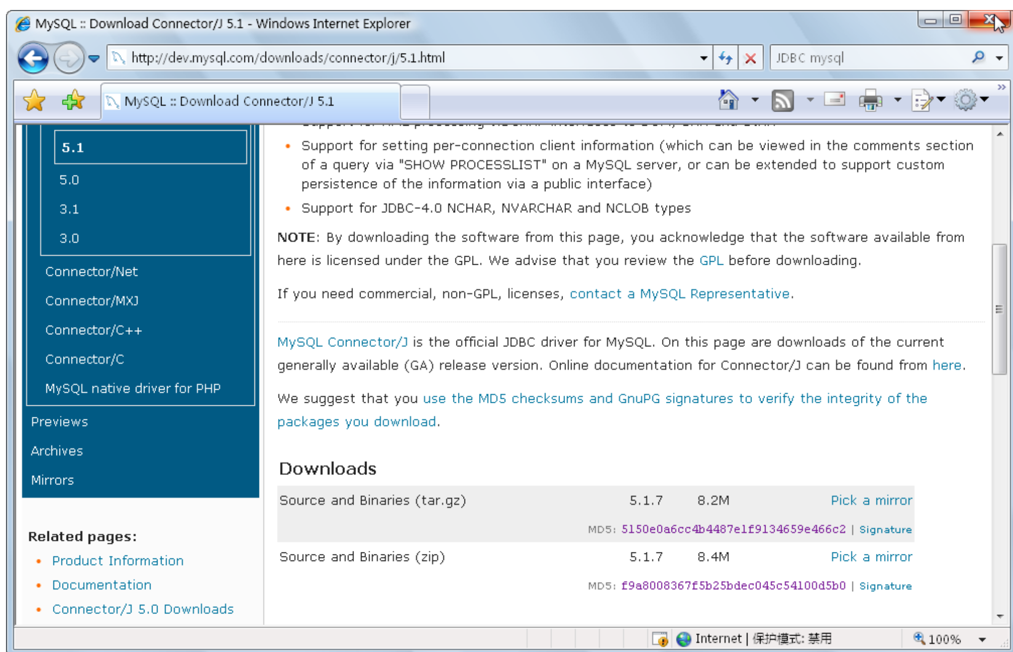


图 14-2 JDBC 驱动

14.3.2　配置 MySQL 驱动

将 MySQL 驱动下载后进行解压，寻找一个名为 mysql-connector-java.jar 的文件，找到后将它复制到需要的位置，例如，D 盘根目录下，然后对它进行配置，其具体操作步骤如下。

1）在桌面上单击计算机图标，单击鼠标右键，在弹出的快捷菜单中选择“属性”命令，然后单击左边的“高级系统设置”连接，然后单击“环境变量”按钮，如图 14-3 所示。

2）在前面的章节中曾经讲解过如何配置 JDK，建立了一个名为“CLASSPATH”的环境变量。现在找到这个环境变量，单击“编辑”按钮重新对它进行编辑，如图 14-4 所示。

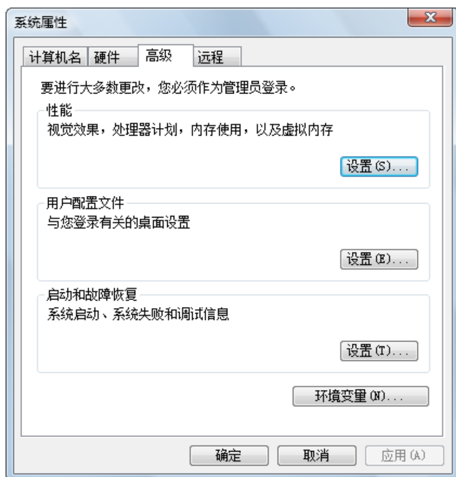


图 14-3　“系统属性”对话框

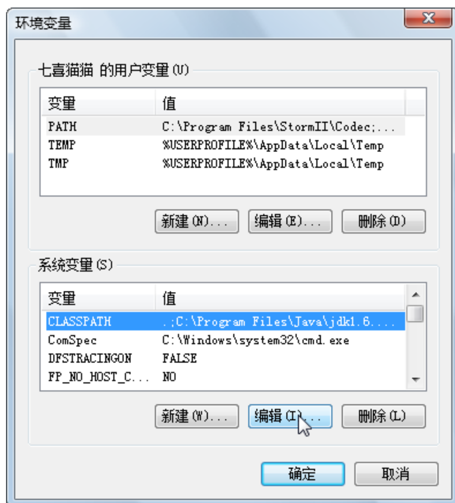


图 14-4　编辑 CLASSPATH

3）在“CLASSPATH”的变量值后面加上分号“;”，再加上“mysql-connector-java.jar”文件的路径。因为笔者放置在了 D 盘根目录，所以笔者添加的是“;D:\mysql-connector-java-5.1.7-bin.jar”，然后单击“确定”按钮，再次单击“确定”按钮，如图 14-5 所示。

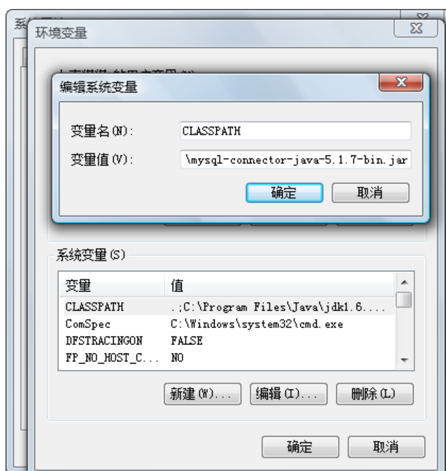


图 14-5　编辑系统变量

14.3.3　通过 Java 程序连接 MySQL

接下来编写一个 Java 程序来连接 MySQL，首先启动 MySQL 管理工具，然后执行如下代码，将数据库导入本地 MySQL 服务器中。

```
-- 数据库: `scutcs`
-- 表的结构 `student`
--
CREATE TABLE IF NOT EXISTS `student` (
  `SNO` char(7) NOT NULL,
```

```
  `SNAME` varchar(8) NOT NULL,
  `SEX` char(2) NOT NULL,
  `BDATE` date NOT NULL,
  `HEIGHT` decimal(5,2) DEFAULT '0.00',
  PRIMARY KEY (`SNO`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- 导出表中的数据 `student`
--
INSERT  INTO  `student`  (`SNO`,  `SNAME`,  `SEX`,  `BDATE`,  `HEIGHT`)
VALUES
('1', 'momo', '8', '2000-06-23', 145.00),
('2', 'gaogao', '7', '2001-05-27', 150.00);
```

接下来编写一个 Java 程序实现连接。假设这个数据库的用户名是 root，密码是 1234，其他的都是默认。这段程序的具体代码如下。

```
import java.sql.*;
public class JDBC1
{
public static void main(String[] args)
{
        // 驱动程序名
        String driver = "com.mysql.jdbc.Driver";
        // URL 指向要访问的数据库名 scutcs
        String url = "jdbc:mysql://127.0.0.1:3306/scutcs";
        // MySQL 配置时的用户名
        String user = "root";
         // MySQL 配置时的密码
        String password = "1234";
        try {
         // 加载驱动程序
         Class.forName(driver);
         // 连续数据库
         Connection  conn  =  DriverManager.getConnection(url,  user,
password);
         if(!conn.isClosed())
          System.out.println("Succeeded connecting to the Database!");
         // statement 用来执行 SQL 语句
         Statement statement = conn.createStatement();
         // 要执行的 SQL 语句
         String sql = "select * from student";
         // 结果集
         ResultSet rs = statement.executeQuery(sql);
         System.out.println("-----------------");
         System.out.println("执行结果如下所示:");
```

```
                System.out.println("----------------");
                System.out.println(" 学号" + "\t" + " 姓名"+"\t"+"出生日期");
                System.out.println("----------------");
                String name = null;
                while(rs.next()) {
                    // 选择 sname 这列数据
                 name = rs.getString("sname");
                    // 首先使用 ISO-8859-1 字符集将 name 解码为字节序列并将结果存储新的
                   // 字节数组中。
                 // 然后使用 GB2312 字符集解码指定的字节数组
                 name = new String(name.getBytes("ISO-8859-1"),"GB2312");
                 // 输出结果
                 System.out.println(rs.getString("sno") + "\t" + name);
                }
                rs.close();
                conn.close();
               } catch(ClassNotFoundException e) {
                System.out.println("Sorry,can`t find the Driver!");
                e.printStackTrace();
               } catch(SQLException e) {
                e.printStackTrace();
               } catch(Exception e) {
                e.printStackTrace();
               }
        }
    }
```

将上述代码执行得到如图 14-6 所示的效果。

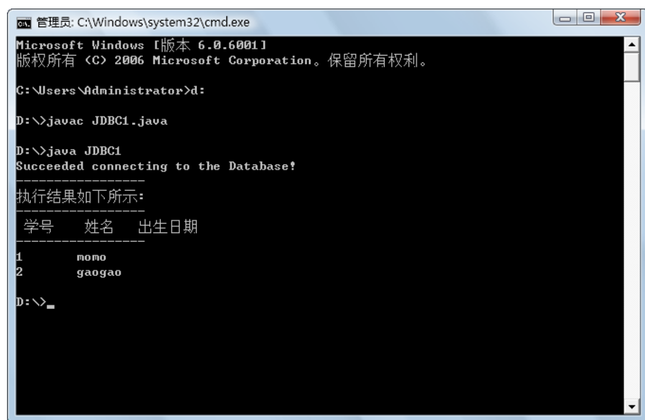


图 14-6　连接数据库

14.3.4　在 Eclipse 中配置驱动

在真正的开发过程中，很少人会使用前面一节讲解的方式开发。最高效的做法是使用

Eclipse 或者 MyEclipse 开发 Java 程序。Eclipse 和 MyEclipse 的驱动配置方法是一样的，下面以 Eclipse 为例进行配置，其具体操作步骤如下。

1）启动 Eclipse，然后选择下载的驱动文件，单击鼠标右键选择复制命令，然后在 Eclipse 里选择需要的项目，如图 14-7 所示。

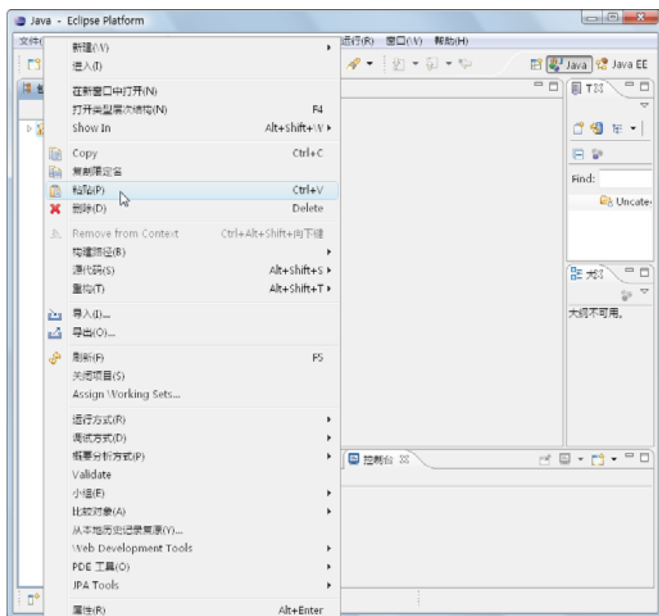


图 14-7　复制与粘贴

2）选择加载的驱动，然后单击鼠标右键，在弹出的快捷菜单中选择“构建路径/添加至构建路径”命令，如图 14-8 所示。

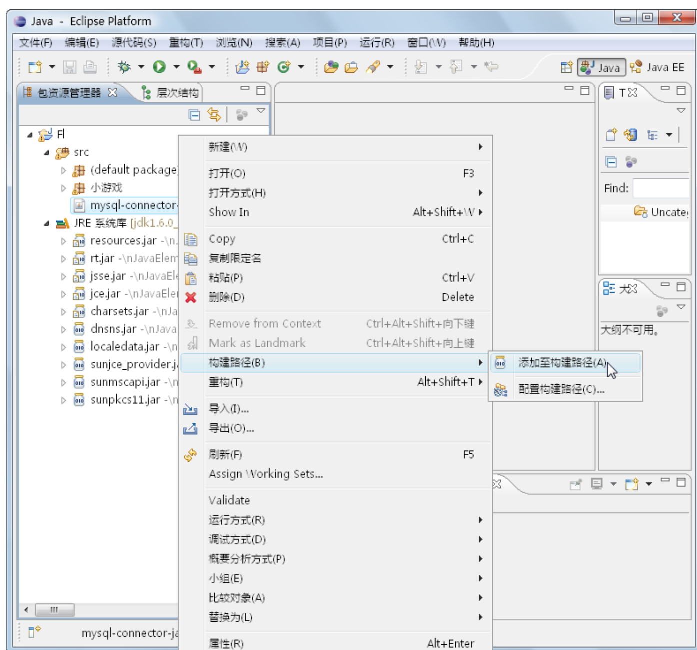


图 14-8　选择命令

14.3.5　在 Eclipse 中测试是否连接成功

在 Eclipse 中可以测试前面编写的文件 JDBC1.java，在 Eclipse 中运行后将会得到如图 14–9 所示的效果。

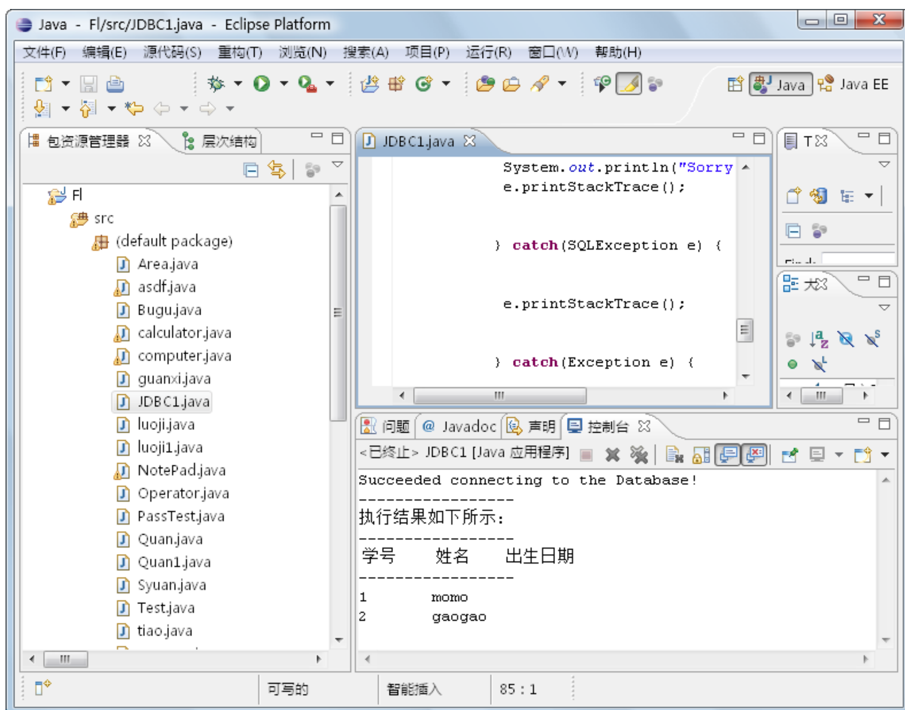


图 14–9　在 Eclipse 中配置的结果

提示：在上面的程序中，连接 MySQL 数据库的方法很简单。在实际开发过程中，连接数据库后肯定需要操作数据库，所以在上述程序中有操作数据库的代码。下面是通用的连接 MySQL 数据库的代码。

```
        // 驱动程序名（连接数据库的第一步）
                String driver = "com.mysql.jdbc.Driver";
                // URL 指向要访问的数据库名 scutcs（连接数据库的第二步）
                String url = "jdbc:mysql://127.0.0.1:3306/scutcs";
                // MySQL 配置时的用户名
                String user = "root";
                // MySQL 配置时的密码
                String password = "****";
                try {
                 // 加载驱动程序
                 Class.forName(driver);
                 // 连续数据库
                 Connection  conn  =  DriverManager.getConnection(url,  user,
password);
```

14.4　连接 SQL Sever 2000

连接 SQL Sever 2000 的方法十分简单，和连接 MySQL 的方法有点相似，其连接格式如下。

```
        //加载驱动
        Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver").newInsta
nce();
        //mydb 为数据库
        String
url="jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=mydb";
        //用户名
        String user="sa";
        //密码
        String password="";
        Connection conn= DriverManager.getConnection(url,user,password);
```

14.4.1　安装 SQL Server 2000 的 JDBC 驱动

可以在网络中获取 SQL Server 2000 Driver for JDBC SP3 驱动程序，下载后需要将其安装。安装过程十分简单，具体安装步骤如下。

1）双击下载的 SQL Server 2000 Driver for JDBC SP3.exe，经过短暂的等待后会打开如图 14-10 所示的对话框。

2）单击“Next”按钮后弹出协议对话框，在此选择第一个单选按钮，即同意协议内容，然后单击“Next”按钮，如图 14-11 所示。

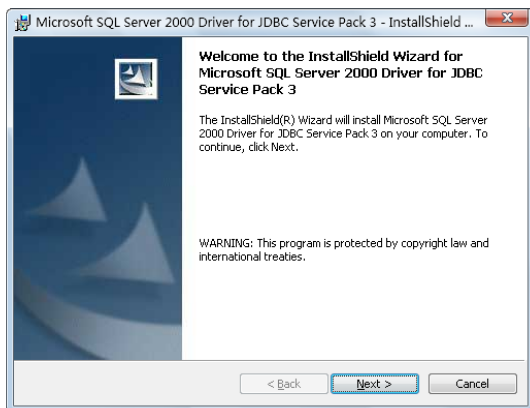


图 14-10　第一个向导



图 14-11　协议对话框

3）在打开的对话框中按照默认设置即可，然后单击“Next”按钮，如图 14-12 所示。

4）在打开的对话框中单击“Install”按钮后即可进行安装，安装完成后会打开一个完成安装的对话框，此时单击“Finish”按钮，如图 14-13 所示。

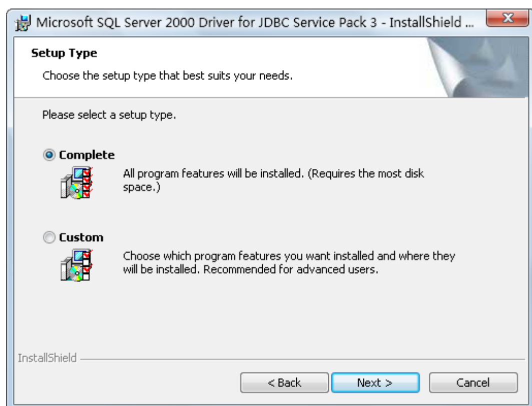


图 14-12　按照默认设置

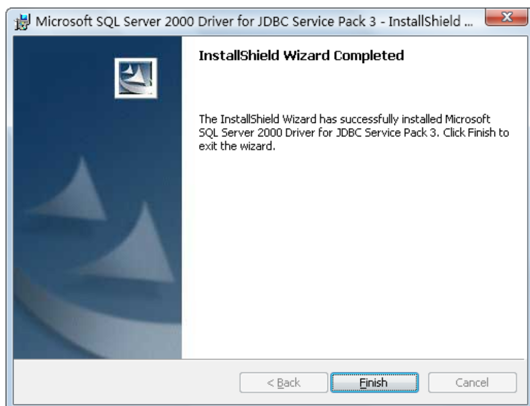


图 14-13　完成安装

14.4.2　配置 SQL Server 2000 的 JDBC 驱动

安装完成后需要查看安装文件夹的文件，然后进行配置。配置 SQL Server 2000 的 JDBC 驱动的具体操作步骤如下。

1）打开安装文件夹中的“lib”文件夹，此时可以查看到 lib 文件夹中的内容，如图 14-14 所示。

2）在桌面上单击计算机图标，单击鼠标右键，在弹出的快捷菜单中选择“属性”命令，然后单击左边的“高级系统设置”连接，然后单击“环境变量”按钮，如图 14-15 所示。

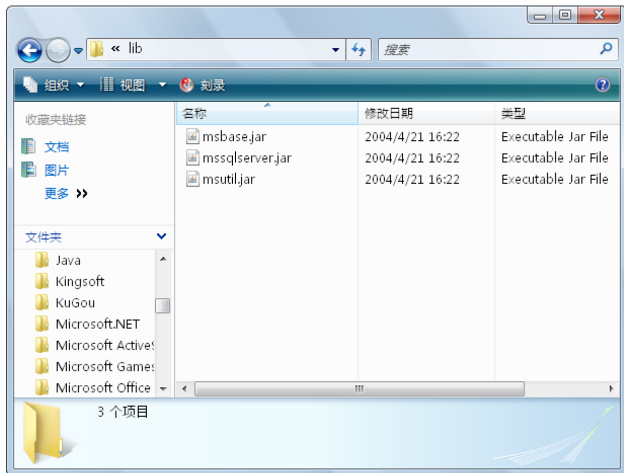


图 14-14　lib 文件夹的 3 个文件

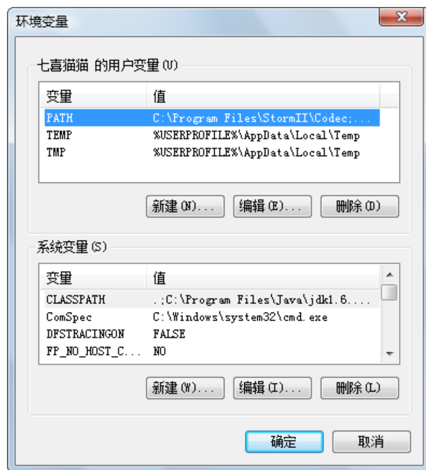


图 14-15　系统属性窗口

3）在前面的章节中曾经讲过配置 JDK 的方法，曾经建立了一个名为“CLASSPATH”的环境变量。现在要找到这个环境变量，单击“编辑”按钮重新对它进行编辑，如图 14-16 所示。

4）在“CLASSPATH”的环境变量值后面加上分号“;”，然后再加上“mysql-connector-java.jar”的路径，在这里要根据自己计算机设置的环境变量位置来设置，例如，笔者的计算机设置为“C:\Program Files\Microsoft SQL Server 2000 Driver for JDBC\lib\msbase.jar;

C:\Program Files\Microsoft SQL Server 2000 Driver for JDBC\lib\mssqlserver.jar;C:\Program Files\Microsoft SQL Server 2000 Driver for JDBC\lib\msutil.jar”。然后单击“确定”按钮，再次单击“确定”按钮，如图 14-17 所示。

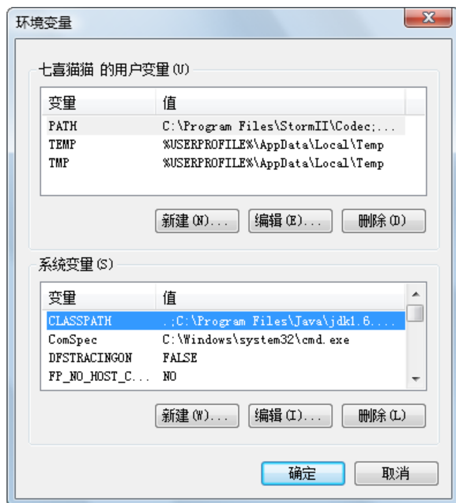


图 14-16　设置环境变量

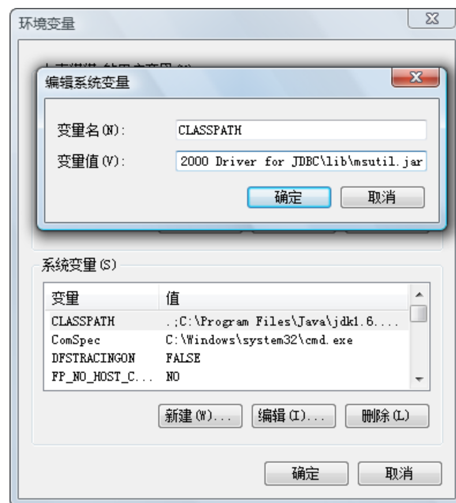


图 14-17　设置 CLASSPATH

14.4.3　连接数据库

配置完成后可以编写一段简单的代码来连接 SQL Sever 2000 数据库，例如，可以用如下代码连接 SQL Sever 2000 中自带的数据库 pubs。

```
        import java.sql.*;
        public class TestDB
        {
        public static void main(String[] args)
        {

        String driverName = "com.microsoft.jdbc.sqlserver.SQLServerDriver";
        //加载驱动
        String  dbURL  =  "jdbc:microsoft:sqlserver://localhost:1433;  Database
Name=pubs";
        //连接数据库
        String userName = "sa"; //服务器用户名
        String userPwd = ""; //用户名密码
        Connection dbConn;
        try
         {
        Class.forName(driverName);
        dbConn = DriverManager.getConnection(dbURL, userName, userPwd);
        System.out.println("连接成功");
        }
        catch (Exception e)
```

```
    {
    e.printStackTrace();
    }
    }
    }
```

将上述代码编译后会得到如图 14-18 所示的结果。

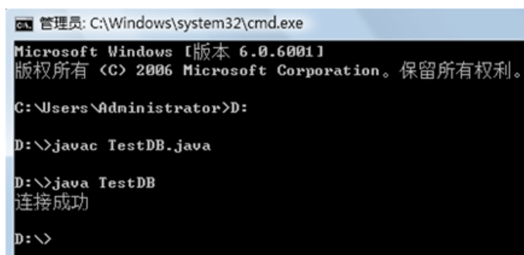


图 14-18　连接 SQL Sever 2000 成功

14.4.4　在 Eclipse 中配置驱动

在安装的文件夹“lib”目录中复制 3 个 JAR 文件，然后对它进行配置。在 Eclipse 中配置驱动的基本流程如下。

1）启动 Eclipse，然后选择下载的驱动文件，单击鼠标右键选择复制命令，然后在 Eclipse 里选择需要的项目，如图 14-19 所示。

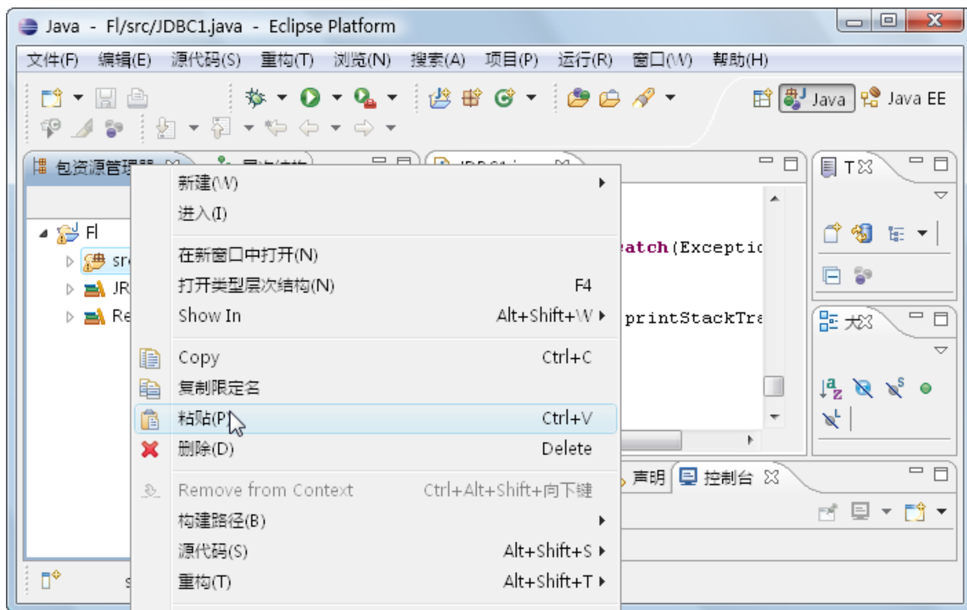


图 14-19　粘贴 JAR 文件

2）选择加载的驱动，然后单击鼠标右键，在弹出的快捷菜单中选择“构建路径/添加至构建路径”命令，如图 14-20 所示。

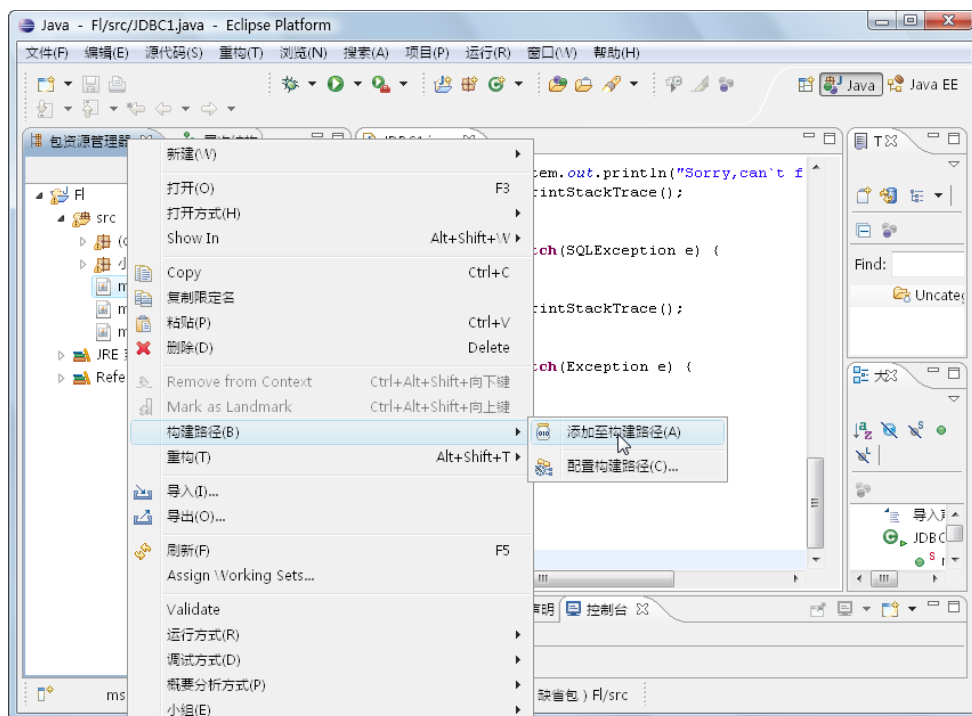


图 14-20　添加到构建路径

3）用相同的方法分别添加 3 个 JAR 文件，最后得到如图 14-21 所示的结果。

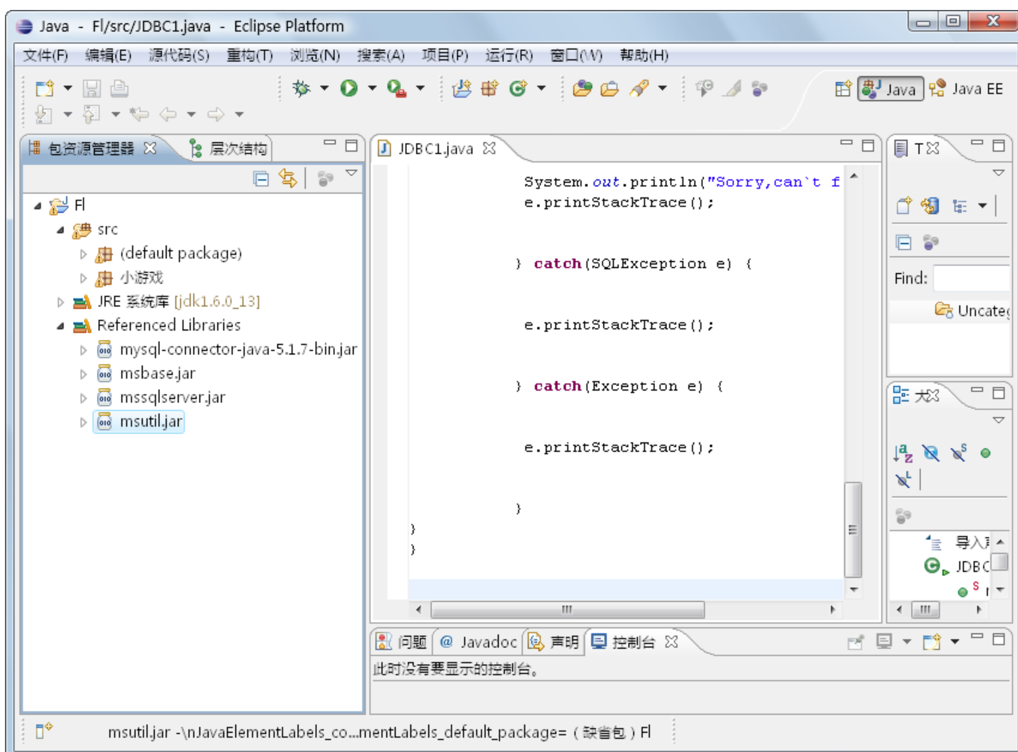


图 14-21　添加 3 个 JAR 文件

14.4.5　在 Eclipse 中测试是否连接成功

接下来使用前面编写的文件 TestDB.java，在 Eclipse 中运行后将会得到如图 14–22 所示的效果。

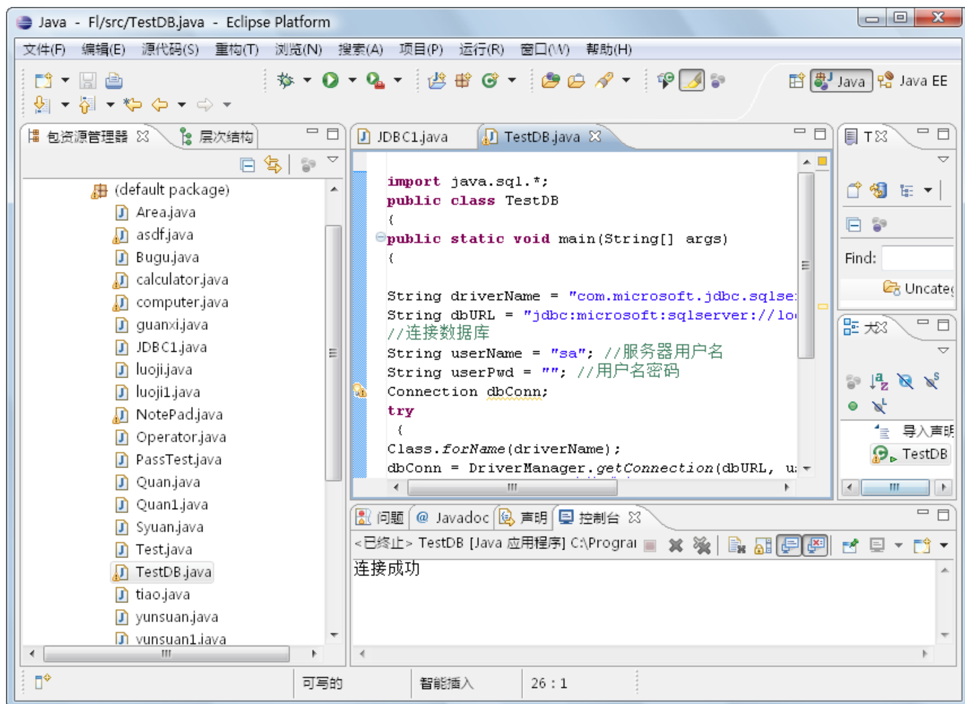


图 14–22　在 Eclipse 中配置驱动

14.5　连接其他数据库的方法

在前面讲解了连接 MySQL 数据库的方法，实际上在编程过程中还可能需要连接不同的数据库，接下来将介绍连接主流数据库的基本方法。

1）连接 Oracle 8/8i/9i 数据库的通用代码如下。

```
//加载驱动
Class.forName("oracle.jdbc.driver.OracleDriver").newInstance();
//orcl 为数据库的 SID
String url="jdbc:oracle:thin:@localhost:1521:orcl";
//用户名
String user="test";
//密码
String password="test";
Connection conn= DriverManager.getConnection(url,user,password);
```

2）连接 DB2 数据库的通用代码如下。

```
//加载驱动
Class.forName("com.ibm.db2.jdbc.app.DB2Driver ").newInstance();
// sample 为数据库的 SID
String url="jdbc:db2://localhost:5000/sample";
//用户名
String user="admin";
//密码
String password="";
Connection conn= DriverManager.getConnection(url,user,password);
```

3）连接 Sybase 数据库的通用代码如下。

```
//加载驱动
Class.forName("com.sybase.jdbc.SybDriver").newInstance();
//myDB 为你的数据库名
String url =" jdbc:sybase:Tds:localhost:5007/myDB";
//用户和密码
Properties sysProps = System.getProperties();
SysProps.put("user","userid");
SysProps.put("password","user_password");
Connection conn= DriverManager.getConnection(url, SysProps);
```

4）连接 PostgreSQL 数据库的通用代码如下。

```
//加载驱动
Class.forName("org.postgresql.Driver").newInstance();
//myDB 为数据库名
String url ="jdbc:postgresql://localhost/myDB"
//用户名
String user="myuser";
//密码
String password="mypassword";
Connection conn= DriverManager.getConnection(url,user,password);
```

14.6　JDBC 常用接口

在本节的内容中，将详细讲解 JDBC 中常用接口的基本知识，目的是为读者学习本书后面的知识打下坚实的基础。

14.6.1　DriverManager 类

接口 DataSource 是在 JDBC 2.0 API 中的新增内容，它提供了连接到数据源的另一种方法，连接到数据源的首选方法是使用 DataSource 对象。作为初始化工作的一部分，类 DriverManager 会尝试加载在“jdbc.drivers”系统属性中引用的驱动程序类。类 DataSource 类中的常用方法如下。

- ❑ getConnection（String url, String user, String password）：建立到给定数据库 URL 的连

接，DriverManager 会从已注册的驱动程序集中选择一个适当的驱动程序。

- setLoginTimeout（int seconds）：设置驱动程序试图连接到某一数据库时将等待的最长时间，单位为 s。
- getLoginTimeout()：获得驱动程序试图登录到某一数据库时可以等待的最长时间，单位为 s。
- println（String message）：将一条消息打印到当前 JDBC 记录流中。

14.6.2 Connection 接口

接口 Connection 能够实现与特定数据库的连接，并在连接上下文中执行 SQL 语句并返回结果。Connection 对象数据库可以提供描述表、所支持的 SQL 语法、存储过程和此连接的信息。这些信息是使用 getMetaData()方法获得的。Connection 接口中各个方法的介绍如下。

- createStatement()：创建一个 Statement 对象将 SQL 语句发送到数据库。
- prepareStatement（String sql）：创建一个 PreparedStatement 对象将参数化的 SQL 语句发送到数据库。带有 IN 参数或不带有 IN 参数的 SQL 语句都可以被预编译并存储在 PreparedStatement 对象中。然后可以有效地使用此对象来多次执行该语句。
- prepareCall（String sql）：创建一个 CallableStatement 对象来调用数据库存储过程。
- setAutoCommit（boolean autoCommit）：将此连接的自动提交模式设置为给定状态。如果连接处于自动提交模式，则将执行其所有 SQL 语句，并将这些语句作为单独的事务提交。否则，其 SQL 语句将成组地进入通过调用 commit 方法或 rollback 方法终止的事务中。默认情况下，新的连接处于自动提交模式。
- getAutoCommit：检索此 Connection 对象的当前自动提交模式。

14.6.3 Statement 接口

接口 Statement 的功能是提供使用 Java 编程语言访问并处理存储在数据源（通常是一个关系数据库）中数据的 API。Statement 接口中的常用方法如下。

- executeQuery（String sql）：执行给定的 SQL 语句，返回单个 ResultSet 对象。
- executeUpdate（String sql）：执行给定的 SQL 语句，该语句可能为 INSERT、UPDATE 或 DELETE 语句，或者不返回任何内容的 SQL 语句。
- clearBatch()：清空此 Statement 对象的当前 SQL 命令列表。
- execute（String sql, int autoGeneratedKeys）：执行给定的 SQL 语句，该语句可能返回多个结果，并通知驱动程序所有自动生成的键都应该可用于获取。如果该 SQL 语句不是 INSERT 语句，或者不是可以返回自动生成键的 SQL 语句，则驱动程序将忽略此信号。

14.6.4 PreparedStatement 接口

接口 PreparedStatement 用于表示预编译的 SQL 语句的对象。SQL 语句被预编译并存储在 PreparedStatement 对象中，这样就可以使用此对象来多次高效地执行该语句。PreparedStatement 接口中的常用方法如下。

- ❑ executeQuery()：在此 PreparedStatement 对象中执行 SQL 查询，并返回该查询生成的 ResultSet 对象。
- ❑ executeUpdate()：在 PreparedStatement 对象中执行 SQL 语句，该语句必须是一个 SQL 数据操作语言（Data Manipulation Language，DML）语句，例如，INSERT、UPDATE 或 DELETE；或者是无返回内容的 SQL 语句，例如，DDL 语句。
- ❑ setInt()：将指定参数设置为给定的 Java int 值。当将此值发送到数据库时，驱动程序将它转换成一个 SQL INTEGER 值。
- ❑ setShort()：将指定参数设置为给定的 Java short 值。当将此值发送到数据库时，驱动程序将它转换成一个 SQL SMALLINT 值。
- ❑ setLong()：将指定参数设置为给定的 Java long 值。当将此值发送到数据库时，驱动程序将它转换成一个 SQL BIGINT 值。
- ❑ setFloat()：将指定参数设置为给定的 Java real 值。当将此值发送到数据库时，驱动程序将它转换成一个 SQL FLOAT 值。
- ❑ setDouble()：将指定参数设置为给定的 Java double 值。当将此值发送到数据库时，驱动程序将它转换成一个 SQL DOUBLE 值。
- ❑ setString()：将指定参数设置为给定的 Java String 值。当将此值发送给数据库时，驱动程序将它转换成一个 SQL VARCHAR 或 LONGVARCHAR 值（取决于该参数相对于驱动程序在 VARCHAR 值上的限制的大小）。
- ❑ setDate()：使用运行应用程序的虚拟机的默认时区，将指定参数设置为给定的 java.sql.Date 值。当将此值发送到数据库时，驱动程序将它转换成一个 SQL DATE 值。

14.6.5 CallableStatement 接口

接口 CallableStatement 能够执行 SQL 存储过程的接口。在 JDBC API 中提供了一个存储过程的 SQL 转义语法，这个语句允许对所有 RDBMS 使用标准方式调用存储过程。此转义语法有两种形式。

1）包含结果参数的形式：如果使用结果参数，则必须将其注册为 OUT 参数。

2）不包含结果参数的形式：可用于输入、输出或同时用于二者。

参数是根据编号按顺序引用的，第一个参数的编号是 1。CallableStatement 接口中的常用方法如下。

- ❑ registerOutParameter：按顺序位置 parameterIndex 将参数注册为 JDBC 类型 sqlType。所有 OUT 参数都必须在执行存储过程前注册。sqlType 为 OUT 参数指定的 JDBC 类型确定必须用于 get 方法，以读取该参数值的 Java 类型。这种形式的 registerOut Parameter 应该在参数类型为 NUMERIC 或 DECIMAL 时使用。
- ❑ wasNull()：查询最后一个读取的 OUT 参数是否为 SQL NULL。此方法只应该在调用获取方法之后调用。
- ❑ getString：以 Java 编程语言中的 String 形式获取指定的 JDBC CHAR、VARCHAR 或 LONGVARCHAR 参数的值。

14.6.6　ResultSet 接口

接口 ResultSet 用于表示数据库结果集的数据表，通常用执行查询数据库的语句生成。ResultSet 对象指向其当前数据行的光标，最初光标被置于第一行之前。next()方法将光标移到下一行。因为该方法在 ResultSet 对象没有下一行时返回 false，所以可以使用它在 while 循环中迭代结果集。默认的 ResultSet 对象是不可更新的，仅有一个向前移动的光标。所以只能迭代一次，并且只能按从第一行到最后一行的顺序进行迭代。并且可以生成可滚动和/或可更新的 ResultSet 对象。接口中的常用方法如下。

- FETCH_FORWARD：指示将按正向（即从第一个到最后一个）处理结果集中的行。setFetchDirection()方法将此常量用做驱动程序的提示，驱动程序可能忽略它。
- FETCH_REVERSE：该常量指示将按逆向（即从最后一个到第一个）处理结果集中的行。setFetchDirection()方法将此常量用做驱动程序的提示，驱动程序可能忽略它。
- FETCH_UNKNOWN：该常量指示结果集中的行的处理顺序未知。setFetchDirection 方法将此常量用做驱动程序的提示。
- int FETCH_FORWARD：该常量指示将按正向（即从第一个到最后一个）处理结果集中的行。setFetchDirection()方法将此常量用做驱动程序的提示。
- getAsciiStream：以 ASCII 字符流的形式获取此 ResultSet 对象的当前行中指定列的值，然后可以按照块从流中读取值。此方法尤其适合于获取很大的 LONGVARCHAR 值。JDBC 驱动程序将执行从数据库格式到 ASCII 的任何必要转换。

14.7　疑难问题解析

在本章的内容中，详细讲解了使用 JDBC 连接数据库的基本知识。本节中，将对本章中比较难理解的问题进行讲解。

读者疑问：在这一章讲解了 Java 连接数据库，并没有讲解 Java Web 连接数据库，怎么样让 Java Web 连接数据库呢？

解答：很简单的，在 Java Web 中，通常调用 Class 文件进去，用户可以使用 JavaBean 的方法连接数据库，这是不矛盾的，在开发过程中，基本没有人把连接数据库的程序写在网页页面中，而是写在 Java 源文件里。只有这样，Java 网页文件才更安全和可靠。

读者疑问：Java Web 连接 SQL Sever 数据库有什么诀窍吗？它是怎么样连接的呢？

解答：连接 SQL Sever 数据库十分简单，加载后驱动，只需编写一段代码即可实现连接，其代码如下。

```
<%@ page contentType="text/html;charset=gb2312"%>
  <%@ page import="java.sql.*"%>
  <html>
  <body>
<
%Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver").newInst
```

```
ance();
          String  url="jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=
pubs";
          //pubs为你的数据库的
          String user="sa";
          String password="";
            Connection conn= DriverManager.getConnection(url,user,password);
            Statementstmt=conn.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);
        </body>
         </html>
```



职场点拨——准备创业

有的程序员不能承受整天加班，有的程序员对朝九晚五的职场生活产生了厌倦，纷纷有了创业的想法。但是创业不能凭借一腔热血，在准备创业时要有很好的心态。如果你已经决定创业了，希望下面的 3 条经验能对你有所帮助。

（1）要永远保持微笑，创业不能依靠偶然。

创业最大的收获是良好的心态，既不生气也不兴奋，必需很平和地对待挫折。如果员工觉得你性格好，就会乐于为你“效力”，做事也会事半功倍。平时要有条不紊地进行创业，用更加成熟的方式去做事。“偶然”的成功，不能够成为人们学习的榜样，我们学习和借鉴的是成功的必然因素。

（2）努力使自己具备一个经营者应有的素质和能力

不论你的规模有多大，都需要面对客户，面对复杂多变的市场，需要应付各种各样的社会关系。我们需要尽量降低经营成本，力求以最少的投入获得最大的经济效益，这就需要经营者具备综合素质和经营能力。企业经营者不是天生的，企业经营者来源于一般工人、农民、基层管理人员，软件公司的经营者大多数来源于程序员。然而，综合素质和经营能力不可能与生俱来，需要在市场经济的实践中，在具体的业务中有目的地不断培养和提高自己。

（3）具备天时、地利、人和

- ❑ 天时：要确保我们开发的产品是顺应潮流的，是大众所需要的。
- ❑ 地利：大家都知道，好的地址，有利于业务的扩大。对于软件公司来说，也同样需要有好的地理位置。
- ❑ 人和：这是最重要的元素，也是不可确定因素中最重要的环节。创业者要用亲和力来感染职员，让他们乐于为团队奉献，乐于充满激情地加班。

第 15 章　数据库编程

在开发 Java Web 程序的过程中，程序员经常需要使用 SQL 语句操作数据库中的信息。SQL 是一门高级的非过程化编程语言，允许用户在高层数据结构上工作。因为 SQL 不要求用户指定对数据的存放方法，也不需要用户了解具体的数据存放方式，所以具有完全不同底层结构的不同数据库系统，可以使用相同的 SQL 语言作为数据输入与管理的接口。通过本章能学到如下知识。

- 数据定义语句。
- 数据查询语句。
- 数据库操作语句。
- 数据库控制语句。
- 数据库中常用函数。
- 职场点拨——客户相处之道。

2010 年 X 月 XX 日，暴雨

今天很郁闷，我做的界面还是没有通过。这个客户真难伺候啊，好几次差点忍不住想骂他一顿……



一问一答

Wisdom：“怎么脸色这么难看，谁惹你了？”

小菜：“这个客户真难伺候啊，否决我的方案好几次了！”

Wisdom：“呵呵，客户是上帝，你得好好把握分寸，本章最后的‘客户相处之道’，希望能对你有所帮助。”

小菜：“嗯，言归正传，数据库也需要编程来实现操作？”

Wisdom：“是啊，客户有多种，数据库操作也同样有多种。常见的数据库操作有添加、删除和更新。种种操作实际上都是用了简单的 SQL 语言。”

15.1　认识操作语句

下面通过一段简单的代码让读者初步体验 SQL 语句的用法，其代码如下。

```
-- phpMyAdmin SQL Dump
-- version 3.1.2
-- http://www.phpmyadmin.net
-- 主机: localhost
-- 生成日期: 2009 年 06 月 22 日 06:35
-- 服务器版本: 5.1.31
-- PHP 版本: 5.2.8
SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
/*!40101 SET @OLD_CHARACTER_SET_CCLUMNT=@@CHARACTER_SET_CCLUMNT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
--
-- 数据库: `scutcs`
--
-- --------------------------------------------------------
--
-- 表的结构 `student`
--
CREATE TABLE IF NOT EXISTS `student` (
  `SNO` char(7) NOT NULL,
  `SNAME` varchar(8) NOT NULL,
  `SEX` char(2) NOT NULL,
  `BDATE` date NOT NULL,
  `HEIGHT` decimal(5,2) DEFAULT '0.00',
  PRIMARY KEY (`SNO`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- 导出表中的数据 `student`
--

INSERT INTO `student` (`SNO`, `SNAME`, `SEX`, `BDATE`, `HEIGHT`) VALUES
('1', 'momo', '8', '2000-06-23', 145.00),
('2', 'gaogao', '7', '2001-05-27', 150.00);
```

通过上述 SQL 语句创建了一个名为 Student 的表，然后录入了两条记录，结果如图 15-1 所示。

在本章接下来的内容中，将深入学习 SQL 语句的各种操作功能，其主要知识点如下。

- 数据定义语言（DDL），例如，CREATE、DROP、ALTER 等语句。
- 数据操作语言（DML），例如，INSERT、UPDATE、DELETE 语句。
- 数据查询语言（DQL），例如，SELECT 语句。
- 数据控制语言（DCL），例如，GRANT、REVOKE、COMMIT、ROLLBACK 等语句。

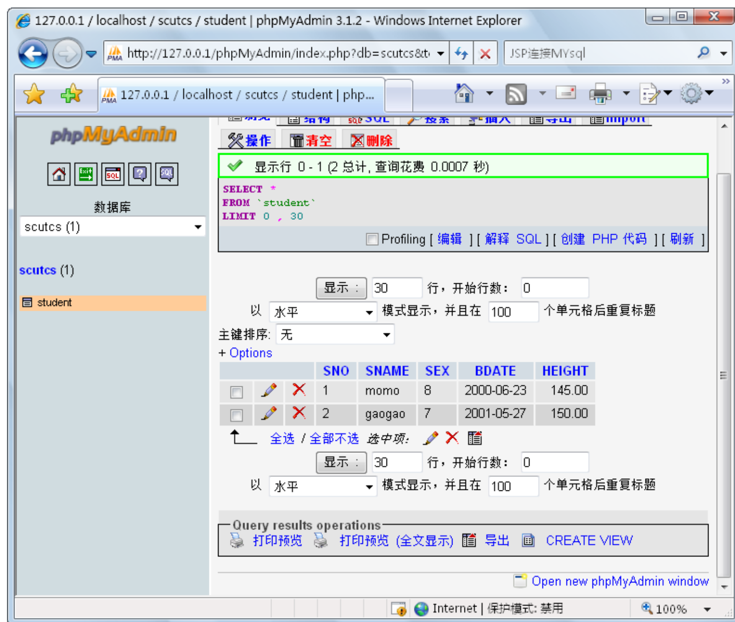


图 15-1　执行结果

15.2　数据定义语句

在使用 SQL 语句操作数据库的过程中，主要实现数据库的查询、新建和删除等操作，下面对上述操作进行详细讲解。

15.2.1　新建数据库和表

新建数据库表很简单，只需要输入如下格式的语句就可以新建一个数据库。

```
CREATE DATABASE `china` ;
```

其参数介绍如下。

- CREATE：关键字。
- DATABASE：数据库新建的关键字。
- China：数据库名。

表格是数据库中储存资料的基本架构。在绝大部分情况下，数据库厂商不可能知道我们是如何储存资料的，所以通常需要自己亲自在数据库中建立表格。虽然市面上有许多数据库工具，可以不需用 SQL 建立表格。在 SQL 中，可以使用 CREATE TABLE 新建一个表，其语法格式如下。

```
CREATE TABLE "表格名"
("栏位 1" "栏位 1 资料种类",
"栏位 2" "栏位 2 资料种类"
... )
```

下面通过一段代码新建一个表格，代码如下。

```
CREATE TABLE customer
(First_Name char(50),
Last_Name char(50),
Address char(50),
City char(50),
Country char(25),
Birth_Date date)
```

执行上面的代码，得到如图 15-2 所示的结果。

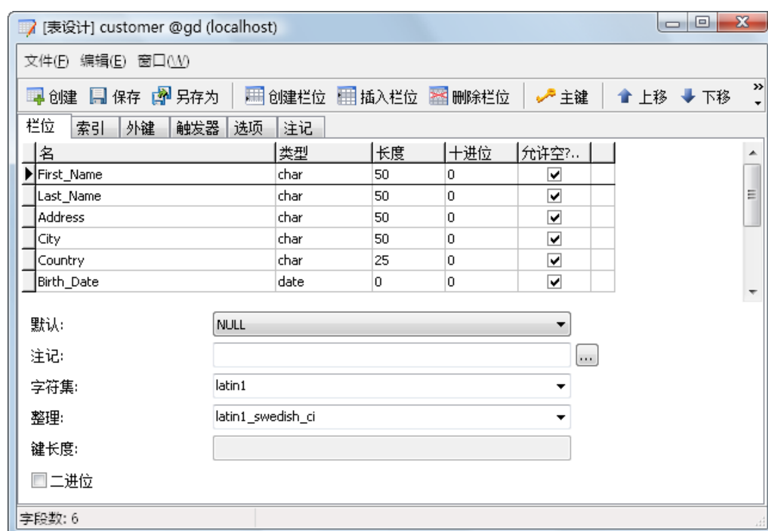


图 15-2　新建表

15.2.2　删除表

有时候根据项目需求需要从数据库中清除一个表格。在 SQL 中可以使用 DROP TABLE 来清除表格。DROP　TABLE 的语法格式如下。

```
DROP TABLE table_name
```

其参数介绍如下。

- DROP：关键字。
- TABLE：数据库新建的关键字。
- table_name：数据库名。

15.2.3　在已有表中进行编辑

ALTER TABLE 语句能够在已有的表中添加、修改或删除列。ALTER TABLE 命令有 3 种用法，这 3 种用法都能简单地操作已有表中的数据。接下来将一一说明这 3 种用法。

1）如需在表中添加列，请使用下列语法。

```
ALTER TABLE table_name
ADD column_name datatype
```

2）要删除表中的列，可以使用下列语法。

```
ALTER TABLE table_name
DROP COLUMN column_name
```



某些数据库系统不允许这种在数据库表中删除列的方式（DROP COLUMN column_ name）。

3）要改变表中列的数据类型，可以使用下列语法。

```
ALTER TABLE table_name
ALTER COLUMN column_name datatype
```

15.3　数据查询语句

在 SQL 语句中经常使用数据库查询功能，接下来将详细讲解如何在数据库中进行查询和查询的一些技巧。

15.3.1　SELECT 语句

SELECT 语句的功能是将资料从数据库中的表格内选出，即从（FROM）数据库中的表格内选出（SELECT）。表格是一个数据库内的结构，它的作用是储存资料。可以使用 SELECT 语句来查询某表格中的数据，其格式如下。

```
SELECT "栏位名" FROM "表格名"
```

假设存在一个表，此表的内容如下所示。

store_name	Sales	Date
Los Angeles	$1500	Jan-05-1999
San Diego	$250	Jan-07-1999
Los Angeles	$300	Jan-08-1999
Boston	$700	Jan-08-1999

如果要选出表中所有的店名（store_Name）信息，只需要输入下面的语句即可实现。

```
SELECT store_name FROM Store_Information
```

执行上述语句，就会得到下面的结果。

store_name

Los Angeles

San Diego
Los Angeles
Boston

15.3.2 选择列表

在接下来的内容中，将讲解 SQL 中使用最多的简单查询的知识。

1．显示所有列

选择并显示所有列这一功能十分常见，使用方法很简单，使用下面的 SQL 语句即可显示表 mmm 中所有列的数据。

```
SELECT *
FROM mmm
```

2．选择部分列并指定它们的显示次序

查询结果集合中数据的排列顺序与选择列表中所指定的列名排列顺序相同，具体格式如下。

```
SELECT nickname,email
FROM mmm
```

3．更改列表

在选择列表中可以重新指定某个列标题，定义格式如下。

```
列标题=列名
列名 列标题
```

如果指定的列标题不是标准的标识符格式时，应使用引号定界符。例如，在下列语句中可以使用汉字来显示列标题。

```
SELECT 昵称=nickname,电子邮件=email
FROM mmm
```

4．删除重复行

在 SELECT 语句中，可以使用 ALL 或 DISTINCT 选项来显示表中符合条件的所有行或删除其中重复的数据行，默认为 ALL。使用 DISTINCT 选项时，对于所有重复的数据行在 SELECT 返回的结果集合中只保留一行。

5．限制返回的行数

可以使用 TOP n [PERCENT]选项来限制返回的数据行数，TOP n 表示返回 n 行，而 TOP n PERCENT 表示 n 是一百分数，指定返回的行数等于总行数的百分之几。下面将通过一段代码讲解限制返回的行数的方法，其代码如下。

```
SELECT TOP 2 *
FROM mmm
SELECT TOP 20 PERCENT *
FROM mmm
```

15.3.3 FROM 语句

通过使用 FROM 子句，可以指定 SELECT 语句所查询的及与查询相关的表或视图。在 FROM 子句中最多可指定 256 个表或视图，它们之间用逗号分隔。

当在 FROM 子句中同时指定多个表或视图时，如果选择列表中存在同名列，这时应使用对象名限定这些列所属的表或视图。假如在 aaa 和 bbb 表中同时存在列 clum，在查询两个表中的 clum 时应使用下面语句格式来加以限定。

```
SELECT username,bbb.clum
FROM aaa,bbb
WHERE aaa.clum=bbb.clum
```

在 FROM 子句中可用以下两种格式为表或视图指定别名。

1）表名 as 别名

2）表名 别名

例如，可以用表的别名格式来表示上面语句。

```
SELECT username,b.clum
FROM aaa a,bbb b
WHERE a.clum=b.clum
```

SELECT 不仅能从表或视图中检索数据，还能够从其他查询语句所返回的结果集合中查询数据。例如，下面的代码。

```
SELECT a.au_fname+a.au_lname
FROM authors a,titleauthor ta
 (SELECT title_id,title
FROM titles
   WHERE ytd_sales>10000
   ) AS t
   WHERE a.au_id=ta.au_id
   AND ta.title_id=t.title_id
```

15.3.4 使用 Where 条件查询

通过 WHERE 子句可以设置查询条件，能够过滤掉不需要的数据行。例如，下面语句可以查询年龄大于 20 的数据。

```
SELECT *
FROM aaa
WHERE age>20
```

WHERE 子句包含了各种条件运算符，具体说明如下。

- 比较运算符（大小比较）：>、>=、=、<、<=、<>、!>、!< 。
- 范围运算符（表达式值是否在指定的范围）：BETWEEN…AND…和 NOT BETWEEN…AND…。

- 列表运算符（判断表达式是否为列表中的指定项）：IN（项 1,项 2……）和 NOT IN（项 1,项 2……)。
- 模式匹配符（判断值是否与指定的字符通配格式相符）：LIKE、NOT LIKE。
- 空值判断符（判断表达式是否为空）：IS NULL、NOT IS NULL。
- 逻辑运算符（用于多条件的逻辑连接）：NOT、AND、OR。

在逻辑逻辑运算符中，又分为以下几种情况。

1）范围运算符例：age BETWEEN 10 AND 30 相当于 age>=10 AND age<=30。

2）列表运算符例：country IN ('Germany','China')。

3）模式匹配符例：常用于模糊查找，用于判断列值是否与指定的字符串格式相匹配。可用于 char、varchar、text、ntext、datetime 和 smalldatetime 等类型查询。

15.3.5 联合查询

通过 UNION 运算符，可以将两个或两个以上的 SELECT 语句的查询结果合并成一个结果集合，这个过程就是联合查询。UNION 的语法格式如下。

```
select_statement
UNION [ALL] selectstatement
 [UNION [ALL] selectstatement][…n]
```

参数的具体说明如下。

1）selectstatement：表示待联合的 SELECT 查询语句。

2）ALL 选项：表示将所有行合并到结果集合中。当不指定该项时，被联合查询结果集合中的重复行将只保留一行。

在进行联合查询时，必须注意以下两点。

1）在联合查询时，查询结果的列标题为第一个查询语句的列标题。因此，要定义的列标题必须在第一个查询语句中定义。当要对联合查询的结果进行排序时，必须使用第一个查询语句中的列名、列标题或列序号。

2）当使用 UNION 运算符时，应保证每个联合查询语句的选择列表中有相同数量的表达式，并且每个查询选择表达式应具有相同的数据类型，或是可以自动将它们转换为相同的数据类型。在自动转换时，对于数值类型，系统将低精度的数据类型转换为高精度的数据类型。在包括多个查询的 UNION 语句中，其执行顺序是自左至右，使用括号可以改变这一执行顺序。语法格式如下。

```
查询1 UNION (查询2 UNION 查询3)
```

15.3.6 通过连接查询

连接运算符的功能是实现对多个表查询。连接是关系数据库模型的主要特点，这也是它区别于其他类型数据库管理系统的一个标志。

在关系数据库管理系统中，建立表时并不必确定各数据之间的关系，只能把一个实体的所有信息存放在一个表中。当检索数据时，通过连接操作查询出存放在多个表中的不同实体

的信息。通过连接操作可以在任何时候增加新的数据类型，这给开发人员带来了很大的灵活性。

可以在 SELECT 语句的 FROM 子句或 WHERE 子句中建立连接，在 Transact−SQL 中推荐使用连接查询。在 SQL−92 标准中所定义的 FROM 子句的连接语法格式如下。

```
FROM join_table join_type join_table
 [ON (join_condition)]
```

各个参数的具体说明如下。

1）join_table：用于指出参与连接操作的表名，连接既可以对同一个表操作，也可以对多个表操作，对同一个表操作的连接又称做自连接。

2）join_type：用于指定连接类型，可分为如下 3 种。

- 内连接：即 INNER JOIN，使用比较运算符进行表间某（些）列数据的比较操作，并列出这些表中与连接条件相匹配的数据行。根据所使用的比较方式不同，内连接又分为等值连接、自然连接和不等连接 3 种。
- 外连接：外连接分为 3 种，分别是左外连接（LEFT OUTER JOIN 或 LEFT JOIN）、右外连接（RIGHT OUTER JOIN 或 RIGHT JOIN）和全外连接（FULL OUTER JOIN 或 FULL JOIN）。与内连接不同的是，外连接不只列出与连接条件匹配的行，还列出左表（左外连接时）、右表（右外连接时）或两个表（全外连接时）中所有符合搜索条件的数据行。
- 交叉连接：即 CROSS JOIN，交叉连接没有 WHERE 子句，能够返回连接表中所有数据行的笛卡儿积，其结果集合中的数据行数等于第一个表中符合查询条件的数据行数乘以第二个表中符合查询条件的数据行数。

3）ON（join_condition）子句：用于指出连接条件，它由被连接表中的列和比较运算符、逻辑运算符等构成。

无论上述哪种连接，都不能实现对 text、ntext 和 image 数据类型列进行直接连接，但可以对这 3 种列进行间接连接。代码如下。

```
SELECT p1.pub_id,p2.pub_id,p1.pr_info
FROM pub_info AS
 p1 INNER JOIN pub_info AS p2
ON DATALENGTH(p1.pr_info)=DATALENGTH(p2.pr_info)
```

1. 内连接

内连接查询操作能够列出与连接条件匹配的数据行，它使用比较运算符比较被连接列的列值。内连接分为如下 3 种。

1）等值连接：在连接条件中使用等于号（=）运算符比较被连接列的列值，在其查询结果中列出了被连接表中的所有列，包括其中的重复列。

2）不等连接：在连接条件使用除等于运算符以外的其他比较运算符比较被连接的列的列值。这些运算符包括>、>=、<=、<、!>、!<和<>。

3）自然连接：在连接条件中使用等于（=）运算符比较被连接列的列值，但它使用选择列表指出查询结果集合中所包括的列，并删除连接表中的重复列。

例如，可以使用下面的等值连接，列出 authors 和 publishers 表中位于同一城市的作者和出版社。

```
SELECT *
FROM authors AS a INNER JOIN publishers AS p
ON a.city=p.city
```

也可以使用自然连接，在选择的列表中删除表 authors 和表 publishers 中重复列（city 和 state）。

```
SELECT a.*,p.pub_id,p.pub_name,p.country
FROM authors AS a INNER JOIN publishers AS p
ON a.city=p.city
```

2．外连接

当采用外连接时，在返回的查询结果集合中不仅包含符合连接条件的行，还包括左表（左外连接时）、右表（右外连接时）或两个边接表（全外连接）中的所有数据行。例如，可以使用如下代码，将论坛内容和作者信息连接起来。

```
SELECT a.*,b.*
FROM luntan LEFT JOIN aaa as b
ON a.username=b.username
```

而在下面的代码中，使用全外连接将 city 表中的所有作者信息，user 表中的所有作者信息，以及他们所在的城市信息连接起来。

```
SELECT a.*,b.*
FROM city as a FULL OUTER JOIN user as b
ON a.username=b.username
```

3．交叉连接

假设在 titles 表中有 6 类图书，而 publishers 表中有 8 家出版社，则下列交叉连接检索到的记录数等于 6×8=48 行。

```
SELECT type,
pub_name
FROM titles CROSS JOIN publishers
ORDER BY type
[Post=0][/Post]
```

15.4　数据库操作语句

在本节的内容中，将详细讲解如何操作 SQL 数据库的知识，主要包括插入记录、修改和删除等操作，为读者步入本书后面知识的学习打下基础。

15.4.1　用 INSERT 语句添加行

INSERT 能够向数据库中插入一条新记录，即向某表中添加行。在 INSERT 语句中，

可以指定以下信息。

- ❑ 要插入的行所在的表。
- ❑ 要为其指定值的列的列表。
- ❑ 要为这些列指定的值的列表。

在添加记录时，需要为主键和其他被定义为 NOT NULL 的列指定值。也可以不为定义为 NULL 的列指定值，在默认情况下，这些列都会被设置为空值。

在为一个表添加记录时，要分几种情况来进行具体操作，分别是省略列的列表、为列指定空值、在列值中使用单引号和双引号、从一个表复制到另一个表的记录。接下来将对这几种情况进行讲解。

1．省略列的列表

当为所有的列都提供值时可以省略列的列表，代码如下。

```
INSERT INTO customers
VALUES (7, 'Jane', 'Green', '01-JAN-1940', '800-555-1216');
```

当省略列的列表时，指定的值顺序必须与 DESCRIBE 命令输出结果中显示的列的顺序一致。

2．为列指定空值

关键字 NULL 可以为数据库中某一列指定一个空值，例如，在下面的代码中，使用 INSERT 语句为列 dob 和 phone 指定的都是空值。

```
INSERT INTO customers
VALUES (8, 'Sophie', 'White', NULL, NULL);
```

当使用查询来查看这一行内容时，不能看到列 dob 和 phone 的值，因为这两列都已经被设置为空值。

```
SELECT *
FROM customers
WHERE customer_id = 8;
CUSTOMER_ID FIRST_NAME LAST_NAME  DOB PHONE
----------- ---------- ---------- --------- ------------
8  Sophie White
```

提示：dob 列和 phone 列值都为空。

3．在列值中使用单引号和双引号

在列值中可以使用单引号和双引号。例如，在下面的代码中，使用 INSERT 语句指定新顾客的姓为 O’Malley。

```
INSERT INTO customers
VALUES (9, 'Kyle', 'O''Malley', NULL, NULL);
```

读者需要注意上述姓中字母 O 后面使用了两个单引号。

在下面的代码中，指定了新产品的名称是 The “Great” Gatsby。

```
INSERT INTO products (
product_id, product_type_id, name, description, price
) VALUES (
13, 1, 'The "Great" Gatsby', NULL, 12.99
);
```

4．从一个表向另外一个表复制行

在 INSERT 语句中，可以使用查询从一个表向另外一个表复制行，而不使用列值。此时要求源表和目标表的列数和列的类型必须匹配。

15.4.2　使用 UPDATE 语句修改行

UPDATE 语句能够修改表中某行的数据，其格式如下。

```
UPDATE 表名称 SET 列名称 = 新值 WHERE 列名称 = 某值
```

假设有一个表 Person，表中的内容见表 15-1。

表 15-1　Person 表

LastName	FirstName	Address	City
Gates	Bill	Xuanwumen 10	Beijing
Nnn		Champs-Elysees	

如果想要更改其中一行的数据，为 lastname 是“Nnn”的人添加 firstname，则可以使用下面的 SQL 语句实现。

```
UPDATE Person SET FirstName = 'Fred' WHERE LastName = 'Nnn'
```

修改后结果见表 15-2。

表 15-2　修改后的 Person 表

LastName	FirstName	Address	City
Gates	Bill	Xuanwumen 10	Beijing
Nnn	Fred	Champs-Elysees	

上述代码只是更改了其中的一个值，如果想一次更改若干个值，可以使用下面的 SQL 语句来更改地址（address），并添加城市名称（city）。

```
UPDATE Person SET Address = 'Zhongshan 23', City = 'Nanjing'
WHERE LastName = 'Nnn'
```

修改后的表见表 15-3。

表 15-3　再次修改后的 Person 表

LastName	FirstName	Address	City
Gates	Bill	Xuanwumen 10	Beijing
Nnn	Fred	Zhongshan 23	Nanjing

15.4.3　删除语句 DELETE

使用 DELETE 语句可以删除表中的某行，其格式如下。

```
DELETE FROM 表名称 WHERE 列名称 = 值
```

假设有一张表，表的内容见表 15−4。

表 15−4　未删除前的表

LastName	FirstName	Address	City
Gates	Bill	Xuanwumen 10	Beijing
Nnn	Fred	Zhongshan 23	Nanjing

如果只删除其中的一行，可以使用下面的语句实现"Fred Nnn" 会被删除。

```
DELETE FROM Person WHERE LastName = 'Nnn'
```

执行上面的 SQL 语句后，会得到如表 15−5 所示的表。

表 15−5　删除操作后的表

LastName	FirstName	Address	City
Gates	Bill	Xuanwumen 10	Beijing

可以在不删除表的情况下删除所有的行，代码如下。

```
DELETE FROM table_name
```

或者：

```
DELETE * FROM table_name
```

15.4.4　使用 ORDER BY 语句进行排序

在数据库开发中，经常需要将表格按照某种规则来排序显示，主要有从小往大（ascending）或是从大往小（descending）。在 SQL 中可以运用 ORDER BY 指令实现上述功能，它的语法格式如下。

```
SELECT "栏位名"
FROM "表格名"
[WHERE "条件"]
ORDER BY "栏位名" [ASC, DESC]
```

各个参数的具体说明如下。

1）[]：表示 WHERE 是一定需要的。如果 WHERE 子句存在的话，它是在 ORDER BY 子句之前。

2）ASC：表示结果会以由小往大的顺序列出。

3）DESC：表示结果会以由大往小的顺序列出。如果两者皆没有被写出的话，默认使用

ASC。

在数据库排序过程中，通常会使用两列进行排序，其语法格式如下。

```
ORDER BY "栏位一" [ASC, DESC], "栏位二" [ASC, DESC]
```

如果对这两个栏位都选择由小往大的话，此子句的显示结果是依据“栏位一”由小往大排。如果表中有多个“栏位一”的值相等，则这几个值将依据“栏位二”进行由小往大的排序。举例来说，如果按照 Sales 栏位的由大往小列出 Store_Information 表格中的资料，见表 15-6。

表 15-6　Store_Information 表

store_name	Sales	Date
Los Angeles	$1500	Jan-05-1999
San Diego	$250	Jan-07-1999
San Francisco	$300	Jan-08-1999
Boston	$700	Jan-08-1999

如果输入下面的 SQL 语句进行排列。

```
SELECT store_name, Sales, Date
FROM Store_Information
ORDER BY Sales DESC
```

表的显示结果见表 15-7。

表 15-7　排列后的表

store_name	Sales	Date
Los Angeles	$1500	Jan-05-1999
Boston	$700	Jan-08-1999
San Francisco	$300	Jan-08-1999
San Diego	$250	Jan-07-1999

15.4.5　使用索引 INDEX

索引（Index）能够帮助用户从表格中快速地找到需要的资料。在表格上建立索引有利于提高系统的效率，一个索引可以涵盖一个或多个栏位，建立索引的语法格式如下。

```
CREATE INDEX "INDEX_NAME" ON "TABLE_NAME" (COLUMN_NAME)
```

假设存在如下表格。

```
TABLE Customer
(First_Name char(50),
Last_Name char(50),
Address char(50),
```

```
City char(50),
Country char(25),
Birth_Date date)
```

如果要在 Last_Name 列上建一个索引，可以通过如下代码实现。

```
CREATE INDEX IDX_CUSTOMER_LAST_NAME
on CUSTOMER (Last_Name)
```

如果要在 Last_Name 这个栏位上建一个索引，可以通过如下代码实现。

```
CREATE INDEX IDX_CUSTOMER_LOCATION
on CUSTOMER (City, Country)
```

索引的命名并没有一个固定的方式，通常是在名称前加一个字首，例如，“IDX_”，来避免与资料库中的其他对象混淆。

15.5　数据库控制语句

在数据库中，除了本章前面介绍的 3 种语句外，数据库的控制语句也是十分重要的，尤其是 GRANT、REVOKE、COMMIT 等语句十分常用。接下来将详细讲解数据库控制语句的基本知识。

15.5.1　GRANT 语句

GRANT 语句的作用是将安全对象的权限授予主体，其语法格式如下。

```
Simplified syntax for GRANT
GRANT { ALL [ PRIVILEGES ] }
     | permission [ ( column [ ,...n ] ) ] [ ,...n ]
     [ ON [ class :: ] securable ] TO principal [ ,...n ]
     [ WITH GRANT OPTION ] [ AS principal ]
```

各个参数的具体说明如下。

1）ALL：一般不推荐使用此选项，保留此选项的作用仅是实现向后兼容，它不会授予所有可能的权限。如果授予 ALL 参数，则相当于授予以下几种权限。

- 如果安全对象为数据库，则 ALL 表示 BACKUP DATABASE、BACKUP LOG、CREATE DATABASE、CREATE DEFAULT、CREATE FUNCTION、CREATE PROCEDURE、CREATE RULE、CREATE TABLE 和 CREATE VIEW。
- 如果安全对象为标量函数，则 ALL 表示 EXECUTE 和 REFERENCES。
- 如果安全对象为表值函数，则 ALL 表示 DELETE、INSERT、REFERENCES、SELECT 和 UPDATE。
- 如果安全对象是存储过程，则 ALL 表示 EXECUTE。
- 如果安全对象为表，则 ALL 表示 DELETE、INSERT、REFERENCES、SELECT 和 UPDATE。

- ❑ 如果安全对象为视图，则 ALL 表示 DELETE、INSERT、REFERENCES、SELECT 和 UPDATE。

2）PRIVILEGES：包含此参数是为了符合 ISO 标准。请不要更改 ALL 的行为。

3）Permission：权限的名称。

4）Column：指定表中将授予其权限的列的名称。需要使用括号“()”。

5）Class：指定将授予其权限的安全对象的类。需要范围限定符“::”。

6）Securable：指定将授予其权限的安全对象。

7）TO principal：主体的名称，可为其授予安全对象权限的主体会随安全对象而异。

8）GRANT OPTION：指示被授权者在获得指定权限的同时还可以将指定权限授予其他主体。

9）principal：指定一个主体，执行该查询的主体从该主体获得授予该权限的权利。

15.5.2 REVOKE 语句

REVOKE 语句的功能是删除以前在当前数据库内的用户授予或拒绝的权限，其语法格式如下。

```
REVOKE { ALL | statement [ ,...n ] }
FROM security_account [ ,...n ]
对象权限：
REVOKE [ GRANT OPTION FOR ]
{ ALL [ PRIVILEGES ] | permission [ ,...n ] }
   {
      [ ( column [ ,...n ] ) ] ON { table | view }
      | ON { table | view } [ ( column [ ,...n ] ) ]
      | ON { stored_procedure | extended_procedure }
      | ON { user_defined_function }
   }
{ TO | FROM }
   security_account [ ,...n ]
[ CASCADE ]
[ AS { group | role } ]
```

参数介绍如下。

1）ALL：用于删除所有适用的权限。对于语句权限，只有 SysAdmin 固定服务器角色成员才可以使用 ALL。对于对象权限，SysAdmin 固定服务器角色成员、db_owner 固定数据库角色成员和数据库对象所有者都可以使用 ALL。

2）Statement：是要删除其权限的授权语句。语句列表可以包括以下内容。

- ❑ CREATE DATABASE。
- ❑ CREATE DEFAULT。
- ❑ CREATE FUNCTION。
- ❑ CREATE PROCEDURE。

- CREATE RULE。
- CREATE TABLE。
- CREATE VIEW。
- BACKUP DATABASE。
- BACKUP LOG。
- N：一个占位符，表示在以逗号分隔的列表内可以重复的项目。
- FROM：指定安全账户列表。

15.6　数据库中常用函数

数据库函数是指当需要分析数据清单中的数值是否符合特定条件时，使用的数据库工作表函数。在不同的数据库中的函数会稍有差别。在本节的内容中，将以 MySQL 数据库为标准，讲解一些常用函数的使用知识。

15.6.1　数学函数

数学函数是用来进行简单计算的函数，如求绝对值等。在数据库编程的过程中，主要有以下常用的数字函数，具体介绍如下。

- ABS（x）：返回 x 的绝对值。
- BIN（x）：返回 x 的二进制值（OCT 返回八进制值，HEX 返回十六进制值）。
- CEILING（x）：返回大于 x 的最小整数值。
- EXP（x）：返回值 e（自然对数的底）的 x 次方
- FLOOR（x）：返回小于 x 的最大整数值。
- GREATEST（x1,x2,...,xn）：返回集合中最大的值。
- LEAST（x1,x2,...,xn）：返回集合中最小的值。
- LN（x）：返回 x 的自然对数。
- LOG（x,y）：返回 x 的以 y 为底的对数。
- MOD（x,y）：返回 x/y 的模（余数）。
- PI()：返回π的值（圆周率）。
- RAND()：返回 0～1 内的随机值，可以通过提供一个参数（种子）使 RAND()随机数生成器生成一个指定的值。
- ROUND（x,y）：返回参数 x 的四舍五入的有 y 位小数的值。
- SIGN（x）：返回代表数字 x 的符号的值。
- SQRT（x）：返回一个数的平方根。
- TRUNCATE（x,y）：返回数字 x 截断为 y 位小数的结果。

15.6.2　聚合函数

聚合函数能够对一组值执行计算并返回单一的值。除 COUNT 函数外，聚合函数忽略空值。聚合函数经常与 SELECT 语句的 GROUP BY 子句一同使用。在数据库编程中的常用聚合函数如下。

- AVG（col）：返回指定列的平均值。
- COUNT（col）：返回指定列中非 NULL 值的个数。
- MIN（col）：返回指定列的最小值。
- MAX（col）：返回指定列的最大值。
- SUM（col）：返回指定列的所有值之和。
- GROUP_CONCAT（col）：返回由属于一组的列值连接组合而成的结果。

15.6.3 字符串函数

因为在数据库中有较多字符串，为此提供了很多字符串函数，常用的字符串函数如下。

- ASCII（char）：返回字符的 ASCII 码值。
- BIT_LENGTH（str）：返回字符串的比特长度。
- CONCAT（s1,s2,...,sn）：将 s1,s2,...,sn 连接成字符串。
- CONCAT_WS（sep,s1,s2,...,sn）：将 s1,s2,...,sn 连接成字符串，并用 sep 字符间隔。
- INSERT（str,x,y,instr）：将字符串 str 从第 x 位置开始，y 个字符长的子串替换为字符串 instr，返回结果。
- FIND_IN_SET（str,list）：分析逗号分隔的 list 列表，如果发现 str，返回 str 在 list 中的位置。
- LCASE（str）或 LOWER（str）：返回将字符串 str 中所有字符改变为小写后的结果。
- LEFT（str,x）：返回字符串 str 中最左边的 x 个字符。
- LENGTH（s）：返回字符串 str 中的字符数。
- LTRIM（str）：从字符串 str 中切掉开头的空格。
- POSITION（substr,str）：返回子串 substr 在字符串 str 中第一次出现的位置。
- QUOTE（str）：用反斜杠转义 str 中的单引号。
- REPEAT（str,srchstr,rplcstr）：返回字符串 str 重复 x 次的结果。
- REVERSE（str）：返回颠倒字符串 str 的结果。
- RIGHT（str,x）：返回字符串 str 中最右边的 x 个字符。
- RTRIM（str）：返回字符串 str 尾部的空格。
- STRCMP（s1,s2）：比较字符串 s1 和 s2。
- TRIM（str）：去除字符串首部和尾部的所有空格。
- UCASE（str）或 UPPER（str）：返回将字符串 str 中所有字符转变为大写字母后的结果。

15.6.4 日期和时间函数

日期和时间函数在数据库的编程过程中十分重要，常用的日期和时间函数如下。

- CURDATE()或 CURRENT_DATE()：返回当前的日期。
- CURTIME()或 CURRENT_TIME()：返回当前的时间。
- DATE_ADD（date,INTERVAL int keyword）：返回日期 date 加上间隔时间 int 的结果（int 必须按照关键字进行格式化），如 SELECT DATE_ADD（CURRENT_DATE, INTE RVAL 6 MONTH）;。
- DATE_FORMAT（date,fmt）：依照指定的 fmt 格式格式化日期 date 值。

- DATE_SUB（date,INTERVAL int keyword）：返回日期 date 加上间隔时间 int 的结果（int 必须按照关键字进行格式化），如 SELECT DATE_SUB（CURRENT_DATE, INTE RVAL 6 MONTH）;。
- DAYOFWEEK（date）：返回 date 所代表的一星期中的第几天（1～7）。
- DAYOFMONTH（date）：返回 date 是一个月的第几天（1～31）。
- DAYOFYEAR（date）：返回 date 是一年的第几天（1～366）。
- DAYNAME（date）：返回 date 的星期名，如 SELECT DAYNAME（CURRENT_DATE）;。
- FROM_UNIXTIME（ts,fmt）：根据指定的 fmt 格式，格式化 UNIX 时间戳 ts。
- HOUR（time）：返回 time 的小时值（0～23）。
- MINUTE（time）：返回 time 的分钟值（0～59）。
- MONTH（date）：返回 date 的月份值（1～12）。
- MONTHNAME（date）：返回 date 的月份名，如 SELECT MONTHNAME（CURRENT_ATE);。
- NOW()：返回当前的日期和时间。
- QUARTER（date）：返回 date 在一年中的季度（1～4），如 SELECT QUARTER（CURRENT_DATE）;。
- WEEK（date）：返回日期 date 为一年中第几周（0～53）。
- YEAR（date）：返回日期 date 的年份（1000～9999）。

15.6.5　加密函数

加密函数主要是为了保密数据库的一些信息，常用的加密函数如下。

- AES_ENCRYPT（str,key）：返回用密钥 key 对字符串 str 利用高级加密标准算法加密后的结果，调用 AES_ENCRYPT 的结果是一个二进制字符串，以 BLOB 类型存储。
- AES_DECRYPT（str,key）：返回用密钥 key 对字符串 str 利用高级加密标准算法解密后的结果。
- DECODE（str,key）：使用 key 作为密钥解密加密字符串 str。ENCRYPT（str,salt）使用 UNIX crypt()函数，用关键词 salt（一个可以唯一确定口令的字符串，就像钥匙一样）加密字符串 str。
- ENCODE（str,key）：使用 key 作为密钥加密字符串 str，调用 ENCODE()的结果是一个二进制字符串，它以 BLOB 类型存储 MD5()计算字符串 str 的 MD5 校验和 PASSWORD（str）返回字符串 str 的加密版本，这个加密过程是不可逆的，和 UNIX 密码加密过程使用不同的算法。
- SHA()：计算字符串 str 的安全散列算法(SHA)校验和。

15.7　疑难问题解析

在本章的内容中，详细讲解了数据库编程的基本知识。本节中，将对本章中比较难理解

的问题进行讲解。

读者疑问：在数据库编程中，是怎么实现让 SQL 语句操作数据库这一功能的？

解答：很好理解，在 Java Web 页面中，有许多信息，当需要向数据库读写信息的时候，就将 Java 的一些类和数据库的 SQL 语言结合起来去操作这个数据库，在读写数据库的内容时，更多的是使用 SQL 语句，SQL 语句常嵌套在 Java Web 文件里或者 Java 文件里。

读者疑问：在数据进行查询时，为什么要等一阵子才得到所查到信息？有什么办法改进一下吗？

解答：在数据库编程的过程中，用户一定要注意，用户可以使用索引查询，这样就可以提高查询速度了，如果你编写一个数据库系统，一定要使用索引（Index），索引可以帮助用户从表格中快速地找到需要的资料。举例来说，假设要在一本园艺书中找如何种植青椒的信息。若这本书没有目录的话，那我们是必须要从头开始读，直到找到有关种直青椒的地方为止，但是有目录，就可以很方便地查找到目录，然后找到需要的信息。



职场点拨——客户相处之道

因为客户就是上帝，所以一定要和客户建立良好的关系，才能在职场中运筹帷幄。希望下面的 5 条建议能给读者带来启示。

1）不管你的资历有多老，而无论客户有多年轻，一律要以尊敬的称呼来称呼客户。不要在跟客户还不熟的情况下直呼其名，或者看人家年纪小就叫对方小张、小刘之类的。

2）不管开发的是网站还是桌面软件，首先得保证质量。不能让客户承担任何可能的风险，如果客户会因为选择你的方案而感受到风险的话，就算你给客户 100 万，客户也不敢收，更不会买你开发的产品。

3）当在客户所在单位时，只要见到与客户同在一个办公室的任何人员，不管是负责这个事的人也好，不负责这个事的人也罢，一律要谦逊。如果你觉得别人不是负责人，就对别人不恭敬，只尊重你想“巴结”的这个人，那么你这样做等于让你所想“巴结”的这个人难看，即使这个事他是负责人，他也不一定敢交给你做。

4）当跟客户发生任何利益关系时，在另一个客户面前要一个不提，否则客户会觉得你不可信。商场中有很多行业潜规则，但是我们一个字也不能说。

5）跟客户成为真正的朋友。这点是最难的，想让客户完全相信你，跟你成为真正的朋友，这几乎不可能。但是只要能做到前面的 4 条，就有成为朋友的可能了。

第四篇　核心技术篇

第 16 章　Struts 2 基础

Struts 1 是全世界第一个发布的 MVC 框架，它是 Craig McClanahan 在 2001 年发布的。该框架推出后，就得到了 Java Web 开发者的拥护，经过长达 6 年时间的锤炼，Struts 1 框架更加成熟、稳定，性能也有了很好的保证。到目前为止，Struts 1 依然是世界上使用最广泛的 MVC 框架。Struts 2 是一个全新的框架，与 Struts 1 相比，有革命性的改进。通过本章能学到如下知识。

- Struts 2 的获得与配置。
- 使用 Struts 2 标签库。
- 编写 Action。
- 职场点拨——谈学习效率。

2010 年 X 月 XX 日，阴

最近工作不是很忙，我决定好好学习 Java Web。但是白天得上班，只能晚上学习。怎样才能利用这有限的时间学习呢？看来我得注意学习效率了……



一问一答

小菜：“有快速学习 Java Web 的捷径吗？”

Wisdom：“学习编程没有捷径，急功近利的结果是‘欲速则不达’。你应该端正学习态度，在本章最后我将重点介绍学习的效率。”

小菜：“言归正传，本章的 Struts 2 有什么用？”

Wisdom：“无论是程序员，还是其他行业的工作人员，无论是工作、办事还是学习，都需要追求效率，高效率会达到事半功倍的效果。同样在 Java 领域，也需要提高程序的运行效率，降低程序对内存的占用。为了提高 Java Web 的效率，市面上推出了形形色色的框架，而 Struts 2 就是其中之一。”

16.1　认识 Struts 2

Struts 2 是一个十分优秀的模块，Strut 2 没有继承 Struts 1 的血统，而是继承了 WebWork 的血统。换句话讲，WebWork 衍生了 Struts 2，而不是 Struts 1 衍生了 Struts 2。因为 Struts 2 是 WebWork 的升级，并不是一个全新的框架，因此稳定性、性能等各方面都有很好的保

证。Struts 2 吸收了 Struts 1 和 WebWork 这两者的优势。下面将通过一段代码来演示 Struts 2 的功能，其代码保存在【光盘：源代码/第 16 章/16-1/】中。实现流程如下。

1）新建一个 login.jsp 页面，其代码如下。

```
<%@ page language="java" import="java.util.*" pageEncoding="GB18030"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<%
   String path = request.getContextPath();
   String basePath = request.getScheme() + "://"
         + request.getServerName() + ":" + request.getServerPort()
         + path + "/";
%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
   <head>
      <title>My JSP 'login.jsp' starting page</title>
   </head>
   <body>
      <s:form action="login.action" method="post">
         <s:textfield name="username" label="username" ></s:textfield>
         <s:textfield name="password" label="password"></s:textfield>
         <s:submit label="submit"></s:submit>
      </s:form>
   </body>
</html>
```

2）修改文件 web.xml，其代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web
Application 2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
   <display-name>FirstStruts2</display-name>
   <filter>
      <filter-name>struts2</filter-name>
      <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</
filter-class>
   </filter>
   <filter-mapping>
      <filter-name>struts2</filter-name>
      <url-pattern>/*</url-pattern>
   </filter-mapping>
   <welcome-file-list>
      <welcome-file>login.jsp</welcome-file>
   </welcome-file-list>
</web-app>
```

3）新建文件 LoginAction.java，其代码如下。

```
package com.xgj;
import com.opensymphony.xwork2.ActionSupport;
public class LoginAction extends ActionSupport {
   private String username;
   private String password;
   public String getUsername()
{
     return username;
   }
   public void setUsername(String username) {
     this.username = username;
   }
   public String getPassword()
 {
     return password;
   }
   public void setPassword(String password)
 {
     this.password = password;
   }
   public String execute() throws Exception
 {
        return SUCCESS;
     }
   }
```

4）在“WEB-INF/classes/”目录下新建文件 struts.xml，其代码如下。

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
   "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
   "http://struts.apache.org/dtds/struts-2.0.dtd">
   <struts>
   <package name="struts2" extends="struts-default">
      <action name="login" class="com.xgj.LoginAction">
         <result name="success">result.jsp</result>
      </action>
   </package>
</struts>
```

运行程序，得到如图 16-1 所示的效果。

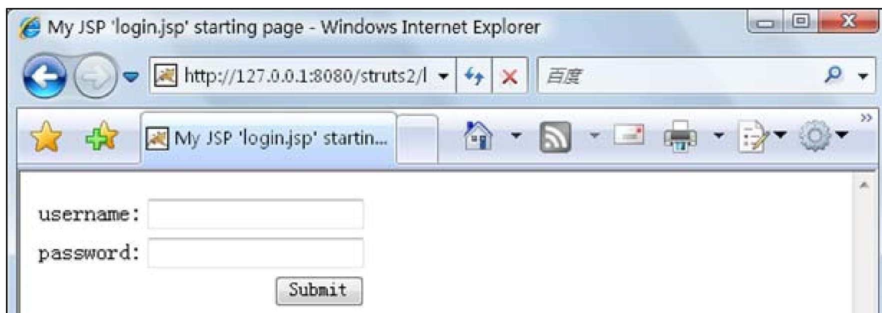


图 16-1　执行结果

如果随意输入用户名和密码，可以得到 16-2 所示的结果。

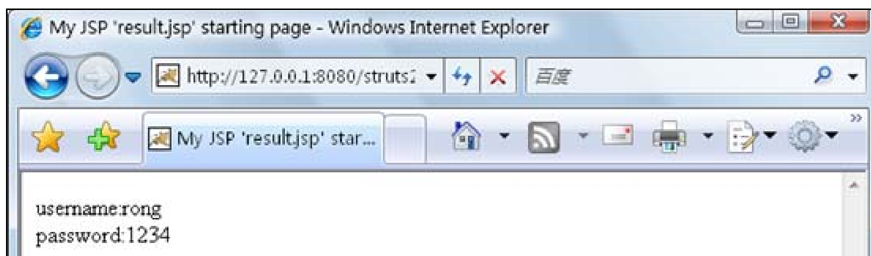


图 16-2　得到的结果

16.2　Struts 2 的思想

在当今应用中，Java 的大部分产品是 B/S（浏览器/服务器）架构的，其中的服务器就是 Web 服务器。由此可见，Web 应用是目前广泛使用的应用模式，而 Struts 2 是一个具有很好的实用价值的 Web MVC 框架。

16.2.1　MVC 思想

MVC 最初存在于 Desktop 程序中，M 是指数据模型，V 是指用户界面，C 则是控制器。使用 MVC 的目的是将 M 和 V 的实现代码分离，从而使同一个程序可以使用不同的表现形式。比如一批统计数据可以分别用柱状图、饼图来表示。C 存在的目的则是确保 M 和 V 的同步，一旦 M 改变，V 应该同步更新。

MVC 即“模型－视图－控制器”，是 Xerox PARC 在 20 世纪 80 年代为编程语言 Smalltalk－80 发明的一种软件设计模式，至今已被广泛使用。最近几年被推荐为 J2EE 平台的设计模式，并且受到越来越多的程序员使用，特别是 ColdFusion 和 PHP 的开发者。

MVC 是一个设计模式，它能够强制性的使应用程序的输入、处理和输出分开。使用 MVC 的应用程序被分成 3 个核心部件，分别是模型、视图、控制器。具体说明如下。

（1）视图

视图是用户看到并与之交互的界面。对于老式的 Web 应用程序来说，视图就是由 HTML 元素组成的界面。在新式的 Web 应用程序中，HTML 依旧在视图中扮演着重要的角

色，但一些新的技术已层出不穷，它们包括 Adobe Flash 和 XHTML，XML/XSL，WML 等一些标识语言和 Web Services。如何处理应用程序的界面变得越来越有挑战性。MVC 的优点是它能为应用程序处理很多不同的视图。在视图中其实没有真正的处理发生，不管这些数据是联机存储的还是一个雇员列表，作为视图来讲，它只是作为一种输出数据并允许用户操纵的方式。

（2）模型

模型表示企业数据和业务规则。在 MVC 的 3 个部件中，模型拥有最多的处理任务。例如，它可能用像 EJB 和 ColdFusion Components 这样的构件对象来处理数据库。被模型返回的数据是中立的，就是说模型与数据格式无关，这样一个模型能为多个视图提供数据。由于应用于模型的代码只需写一次就可以被多个视图重用，所以减少了代码的重复性。

（3）控制器

控制器用于接受用户的输入并调用模型和视图去完成用户的指令。当单击 Web 页面中的超链接和发送 HTML 表单时，控制器本身不输出任何东西和做任何处理，它只是接收请求并决定调用哪个模型构件去处理请求，然后确定用哪个视图来显示模型处理返回的数据。

现在总结 MVC 的处理过程，首先控制器接收用户的请求，并决定应该调用哪个模型来进行处理，然后模型用业务逻辑来处理用户的请求并返回数据，最后控制器用相应的视图格式化模型返回的数据，并通过表示层呈现给用户。

16.2.2　Struts MVC 思想

Struts 是 Apache 组织的一个项目，这是一个开放源码的项目。Struts 是一个比较好的 MVC 框架，它提供了对开发 MVC 系统的底层支持，主要采用 Servlet、JSP 和 customtaglibrary 技术。作为一个 MVC 的框架，Struts 对 Controller、Model、View 和都提供了对应的实现组件。具体说明如下。

（1）Controller

Controller 控制器能够从客户端接受请求，并且选择执行相应的业务逻辑，然后把响应结果送回到客户端。在 Struts 中，Controller 功能是由 ActionServlet 和 ActionMapping 对象构成的。其核心是 Servlet 类型的对象 ActionServlet，它用来接受客户端的请求。

（2）Model

在 Struts MVC 系统中，其 Model 部分可以分为两类，分别是系统的内部状态和改变系统状态的动作。Struts 为 Model 部分提供了 Action 和 ActionForm 对象，所有的 Action 处理器对象都是开发者从 Struts 的 Action 类派生的子类。Action 处理器对象封装了具体的处理逻辑，调用业务逻辑模块，并且把响应提交到合适的 View 组件以产生响应。Struts 提供了 ActionForm 组件对象，它可以通过定义属性描述客户端表单数据。开发者可以从它派生子类对象，利用它和 Struts 提供的自定义标记库结合，可以实现对客户端的表单数据的良好封装和支持。Action 处理器对象可以直接对它进行读/写，而不需要和 request、response 对象进行数据交互。通过使用 ActionForm 组件对象，实现了对 View 和 Model 之间交互的支持。在 Struts 中，建议使用一组 JavaBean 表示系统的内部状态，而根据系统的复杂度也可以使用像 EntityEJB 和 SessionEJB 等组件来实现系统状态。Struts 建议在实现时把“做什么”（Action）和“如何做”（业务逻辑）分离。这样可以实现业务逻辑的重用。

（3）View

可以通过 JSP 技术实现 Struts 应用中的 View 部分。在 Struts 中可以使用自定义的标记，通过这些自定义标记，可以实现和系统中 Model 部分的交互。可以通过使用这些自定义标记创建 JSP 表单，可以实现和 Model 部分中的 ActionForm 的映射，完成对用户数据的封装，同时这些自定义标记还提供了像模板定制等多种显示功能。

16.3　Struts 2 的获得与配置

经过本书前面内容的讲解，想必读者已经明白了 Strust 2 的强大功能。那么应该怎样让自己的计算机也能利用 Struts 2 框架开发 Java Web 程序呢？本节将会一一为读者讲解。

16.3.1　获得 Struts 2

在使用 Struts 2 前必须先下载 Struts 2，Struts 2 是一个免费的 MVC 开发组件。本书中使用的 Strusts 2 版本是 2.1.6，其获取流程如下。

1）打开浏览器，输入网址“http://struts.apache.org/download.cgi”，拖动滑动条，选择需要的版本，如图 16-3 所示。

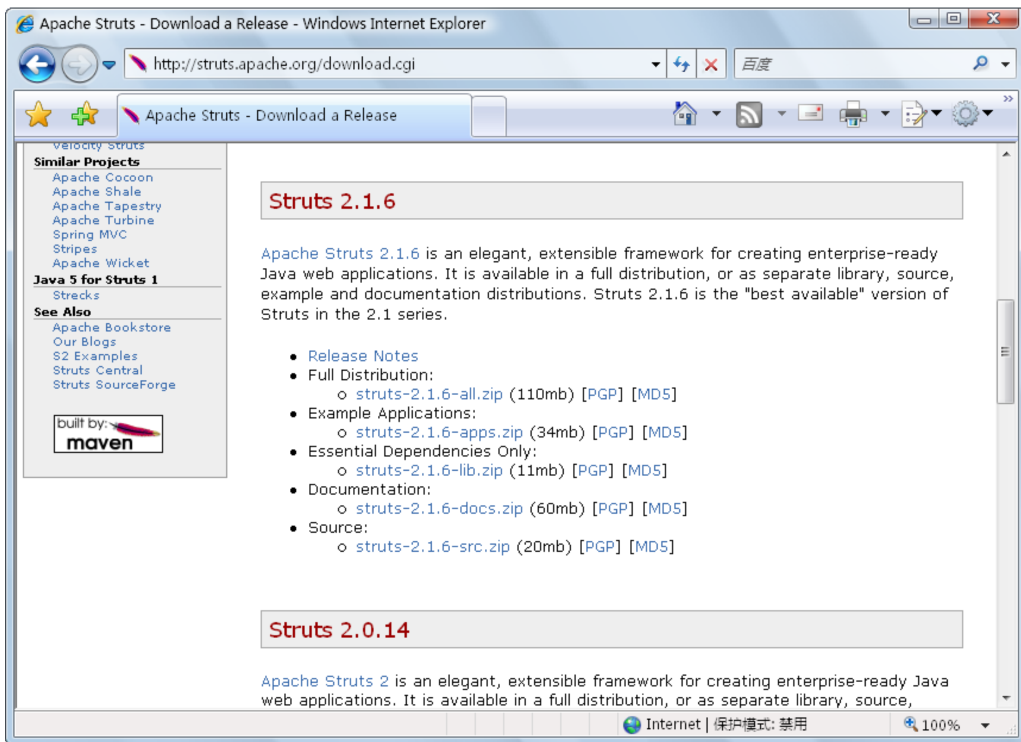


图 16-3　下载页面

2）根据不同的需求选择，在此选择单击 struts-2.1.6-all.zip 超级链接后，将会自动启动一个下载工具，如图 16-4 所示。



图 16-4 下载 Struts 2

提示：在上面的版本中，依次是以下版本，完整版、Struts 2 示例应用、Struts 2 空示例应用、Struts 2 核心库、Struts 2 相关文档、Struts 2 全部代码。

16.3.2 简单配置 Struts 2

下载 Struts 2 后，需要对它进行配置后才能够使用。配置过程十分简单，具体流程如下。

1）将 lib 文件夹下的 Struts2-core-2.1.6.jar、xwork-2.1.2.jar、freemarker-2.3.13.jar 和 ognl-2.6.11.jar 等必需类库复制到 Web 应用的“WEB-INF/lib”路径下。如果 Web 应用需要使用 Struts 2 的更多特性，完全可以将更多的 JAR 文件复制到 Web 应用的“WEB-INF/lib”路径下。如果需要在 DOS 或者 Shell 窗口下手动编译和 Struts 2 相关的程序，则需要将 Struts2-core-2.0.6.jar 和 xwork-2.0.1.jar 添加到系统的 CLASSPATH 环境变量里。

2）编辑 Web 应用的配置文件 web.xml，配置 Struts 2 的核心 Filter。在配置文件 web.xml 中，增加 Struts 2 的核心 Filter 配置。具体代码如下。

```
        <?xml version="1.0" encoding="GBK"?>
        <!-- web-app 是 Web 应用配置文件的根元素，指定 Web 应用的 Schema 信息 -->
        <web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
                         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                           xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.
                           com/xml/ns/j2ee/web-app_2_4.xsd">
                         <!-- 定义 Struts 2 的 FilterDispatcher 的 Filter -->
                          <filter>
                            <!-- 定义核心 Filter 的名字 -->
                          <filter-name>struts2</filter-name>
                            <!-- 定义核心 Filter 的实现类 -->
```

```
                    <filter-class>org.apache.Struts2.dispatcher.FilterDispatcher
                         </ filter-class>
                    </filter>
                   <!-- FilterDispatcher 用来初始化 Struts 2 并且处理所有的
Web 请求 -->
                    <filter-mapping>
                    <filter-name>Struts2</filter-name>
                    <url-pattern>/*</url-pattern>
                    </filter-mapping>
        </web-app>
```

通过上面的步骤，只是在 Web 应用增加了 Struts 2，但是依然没有使用 Struts 2 功能。要想使用它的功能，至少在“WEB-INF/Class”或者“WEB-INF/Lib”目录下增加一个 Struts.xml 文件。

16.3.3 在 MyEclipse 中配置 Strust 2

在开发 Java Web 项目时，一般都是使用 MyEclipse 作为开发工具，下面讲解如何在 MyEclipse 中配置 Strust 2，其具体操作步骤如下。

1）将“lib”文件夹下的 Struts2-core-2.1.6.jar、xwork-2.1.2.jar、freemarker-2.3.13.jar 和 ognl-2.6.11.jar 等必需类库复制，再转到 MyEclipse 窗口，在“Project Explorer”子窗口中的“\WEB-INF\lib”目录中进行粘贴。使用 Eclipse 自动刷新“Project Explorer”子窗口，刚才所粘贴的 Jar 文件应该会出现在“Struts2_HelloWorld\Java Resources: src\Libraries\Web App Libraries”下，如图 16-5 所示。

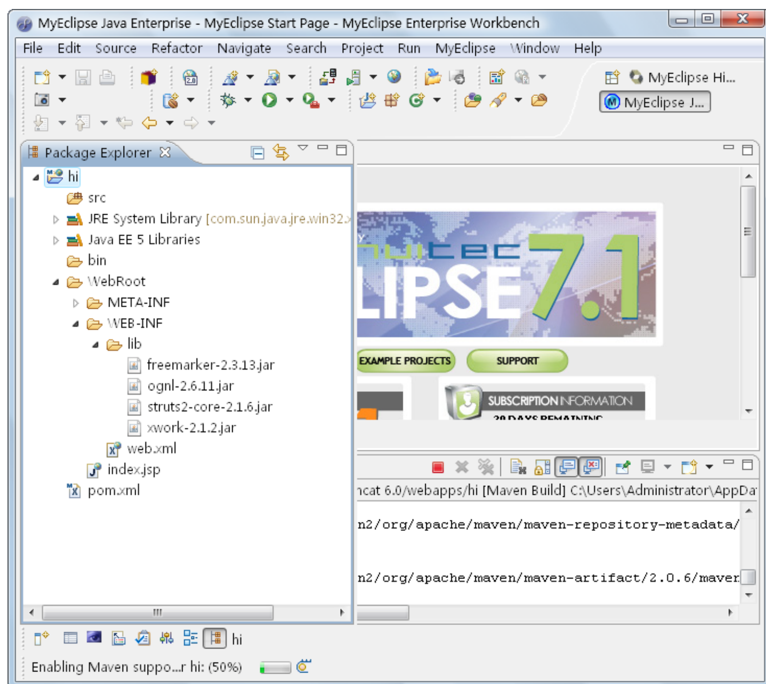


图 16-5 复制需要的文件

2）然后将文件加载到工程中去，如选择第一个 Jar 文件，单击鼠标右键，在弹出的快捷菜单中选择“Build Path/Add to Build Path ”命令，如图 16-6 所示。

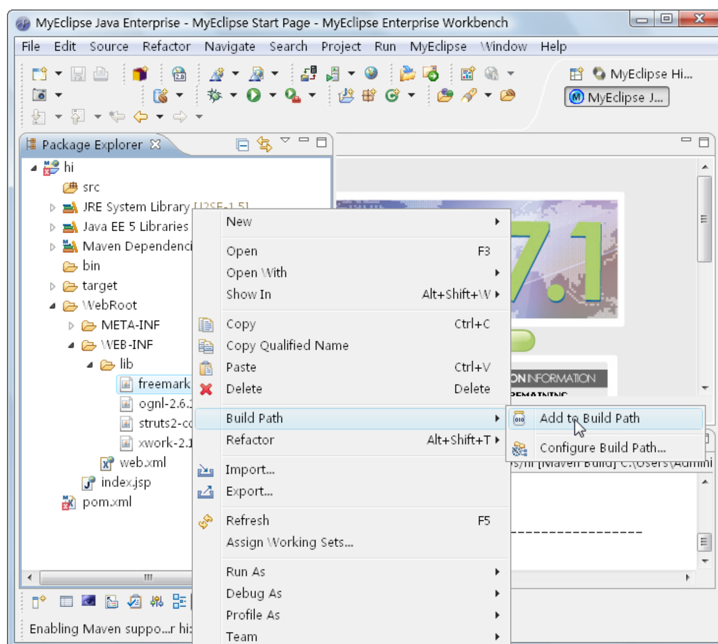


图 16-6　选择命令

3）加载完成后，还需要将 commons-logging-1.0.4.jar 和 commons-fileupload-1.2.1.jar 加载到项目中实现基本的功能，加载完成后的效果如图 16-7 所示。

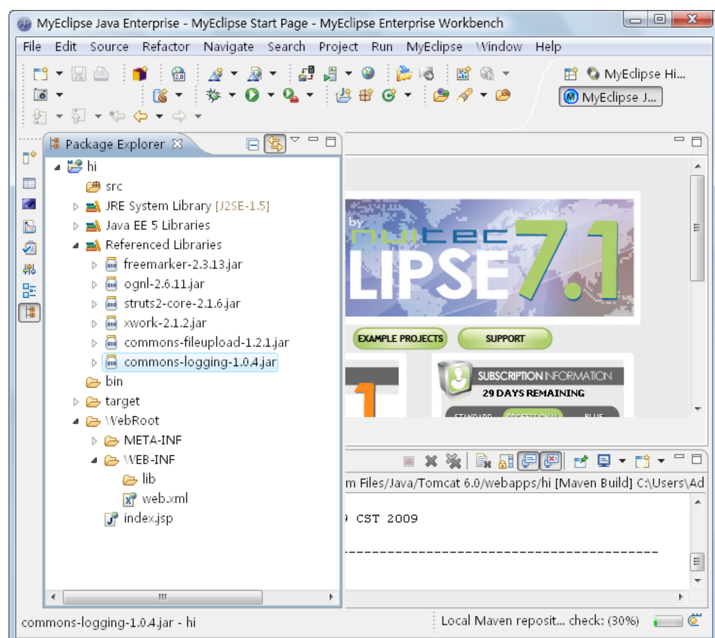


图 16-7　加载 Jar 类文件

16.3.4 运行第一个 Struts 2 的程序

经过上面的配置，就可以根据自己的需要新建一个 Struts 2 的文件，下面将通过一个实例来演示 Struts 2 程序的实现流程。

实例 33：运行 Struts 2 的程序

新建一个简单的测试文件，具体实现流程如下。

1）打开 Web.xml，修改配置参数，修改后的具体代码如下。

```
        <?xml version="1.0" encoding="UTF-8"?>
        <web-app id="WebApp_9" version="2.4" xmlns=http://java.sun.com/xml/ns/j2ee
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
            <display-name>tutorial</display-name>
            <filter>
                <filter-name>struts2</filter-name>
                <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</
filter-class></filter>
            <filter-mapping>
                <filter-name>struts2</filter-name>
                <url-pattern>/*</url-pattern>
            </filter-mapping>
            <welcome-file-list>
                <welcome-file>index.html</welcome-file>
            </welcome-file-list>
        </web-app>
```

2）新建一个名为 HelloWorld.jsp 的页面来呈现信息，具体代码如下。

```
        <% @ taglib prefix="s" uri="/struts-tags" %>
        <html>
            <head>
                <title>Hello World!</title>
            </head>
            <body>
                <h2><s:property value="message" /></h2>
            </body>
        </html>
```

3）新建一个名为 tutorial 的包，然后新建文件 HelloWorld.jsp，具体代码如下。

```
        package tutorial;
        import com.opensymphony.xwork2.ActionSupport;
        public class HelloWorld extends ActionSupport
         {
            public static final String MESSAGE = "Struts is up and running  ";
```

```
        public String execute() throws Exception
        {
                setMessage(MESSAGE);
                return SUCCESS;
            }
            private String message;
        public void setMessage(String message)
         {
                this.message = message;
            }
        public String getMessage()
        {
                return message;
            }
        }
```

4）需要在配置文件中进行相应配置来将两者联系起来。在“src”目录下编辑一个 struts.xml 文件，该文件的具体代码如下。

```
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
    <package name="tutorial" extends="struts-default">
        <action name="HelloWorld" class="tutorial.HelloWorld">
            <result>/HelloWorld.jsp</result>
        </action>
        <!-- Add your actions here -->
    </package>
</struts>
```

将上述代码运行，得到标题栏为“Hello World”、内容为“Struts is up and running!”的页面。


多学一招

上面的代码过于简单，很多内容都未涉及 Struts 2 的知识，只是有了一个 Struts 2 的框架，下面通过几段代码进行讲解，让读者深入地领会 struts 2 的重要性，其代码保存在（光盘：源代码/第 16 章/16-2/）中，具体实现流程如下。

1）在 struts 2 文件夹中复制这个文件，然后将它粘贴到“lib”文件夹中，在 MyEclipse 中的界面效果如图 16-8 所示。

2）修改文件 web.xml，修改后的代码如下。

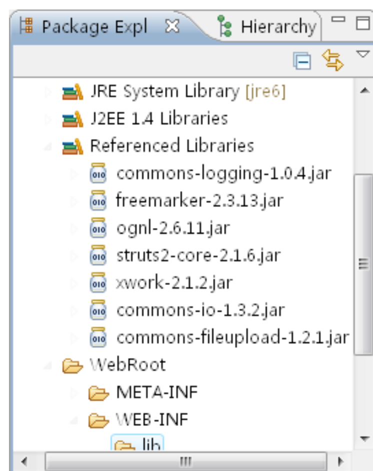


图 16-8　添加的 Jar 文件

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
   <display-name>FirstStruts2</display-name>
   <filter>
      <filter-name>struts2</filter-name>
      <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-class>
   </filter>
   <filter-mapping>
      <filter-name>struts2</filter-name>
      <url-pattern>/*</url-pattern>
   </filter-mapping>
   <welcome-file-list>
      <welcome-file>index.jsp</welcome-file>
   </welcome-file-list>
</web-app>
```

3）新建 FirstAction.java 文件，具体代码如下。

```
package action;
import com.opensymphony.xwork2.ActionSupport;

public class FirstAction extends ActionSupport
{
   private int operand1;
   private int operand2;
     public String execute() throws Exception
   {
      if (getSum() >= 0)  // 如果代数和是非负整数，跳到 positive.jsp 页面
      {
         return "positive";
      }
      else  // 如果代数和是负整数，则跳到 negative.jsp 页面
      {
         return "negative";
      }
   }
    public int getOperand1()
   {
      return operand1;
   }
    public void setOperand1(int operand1)
   {
      System.out.println(operand1);
        this.operand1 = operand1;
```

```
        }
         public int getOperand2()
        {
           return operand2;
        }
        public void setOperand2(int operand2)
        {
           System.out.println(operand2);
           this.operand2 = operand2;
        }
        public int getSum()
        {
           return operand1 + operand2;  // 计算两个整数的代码数和
        }
     }
```

4）新建文件 struts2.xml，具体代码如下。

```
     <?xml version="1.0" encoding="UTF-8" ?>
      <!DOCTYPE struts PUBLIC
         "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
         "http://struts.apache.org/dtds/struts-2.0.dtd">
      <struts>
         <package name="struts2" namespace="/mystruts"
            extends="struts-default">
            <action name="sum" class="action.FirstAction">
               <result name="positive">/positive.jsp</result>
               <result name="negative">/negative.jsp</result>
            </action>
         </package>
      </struts>
```

5）新建文件 sum.jsp，具体代码如下。

```
     <%@ page language="java" import="java.util.*" pageEncoding="GBK" %>
      <%@ taglib prefix="s" uri="/struts-tags"%>
        <html>
         <head>
            <title>输入操作数</title>
         </head>
          <body>

             求代数和
            <br/>
            <s:form action="sum.action" >
               <s:textfield name="operand1" label=" 操作数 1"/>
               <s:textfield name="operand2"  label=" 操作数 2" />
```

```
                <s:submit value="代数和" />
            </s:form>
        </body>
    </html>
```

6）新建文件 positive.jsp，其代码如下。

```
<%@ page language="java" import="java.util.*" pageEncoding="GBK"%>
  <%@ taglib prefix="s" uri="/struts-tags" %>

  <html>
    <head>
      <title>显示代数和</title>
    </head>

    <body>
      代数和为非负整数<h1><s:property value="sum" /></h1>
    </body>
  </html>
```

7）新建一个 negative.jsp 文件，具体代码如下。

```
<%@ page language="java" import="java.util.*" pageEncoding="GBK"%>
  <%@ taglib prefix="s" uri="/struts-tags" %>

  <html>
    <head>
      <title>显示代数和</title>
    </head>

    <body>
      代数和为负整数<h1><s:property value="sum" /></h1>

    </body>
  </html>
```

此时将整个程序部署并预览后会得到一个求代数和的界面，如图 16-9 所示。在表单中输入数字并单击“代数和”按钮后可以实现求和运算。

求代数和

操作数1:

操作数2:

代数和

图 16-9　求代数和

16.4　使用 Struts 2 标签库

在本书前面的内容中，已经学习过 HTML 标签、XML 标记和 JSP 标签。Struts 2 的标签库 Struts 2 Taglib 抽象了这些不同的表示技术。Web 应用程序与传统的 Web 站点有所不同，在 Web 应用程序中可以创建动态的返回。为了更容易在页面上获得动态数据，Struts 2 框架提供了一些标签集，其中有些标签模拟了标准的 HTML 标签，但是提供了一些额外功能。还有一些框架提供了非标准的、但非常有用的控制功能。

16.4.1　Struts 2 标签库的组成

Struts 2 框架的标签库可以分为以下 3 类。

1）用户界面标签（UI 标签）：用来生成 HTML 元素的标签。

2）非用户界面标签：用于实现数据访问和逻辑控制。

3）AJAX 标签：AJAX 是Web 2.0 阶段系列技术和相关产品服务中非常重要的一种技术，其全称是异步 JavaScript 和 XML（即 Asynchronous JavaScript and XML）。该标签能够支持 AJAX 技术。

其中，用户界面标签（UI 标签），可以分为如下两类。

- 表单标签：主要用于生成 HTML 页面的 Form 元素，以及普通表单元素的标签。
- 非表单标签：主要用于生成页面上的 tree、Tab 页等。

非用户界面标签即非 UI 标签，可以分为以下两类。

- 数据访问标签：主要包含用于输出值栈（ValueStack）中的值，完成国际化等功能的标签。
- 流程控制标签：主要包含用于实现分支、循环等流程控制的标签。

提示：通过 Struts 2 标签库，简化了开发代码，增强了代码的可读性和可维护性。

16.4.2　Struts 1 标签与 Struts 2 的区别

熟悉 Struts 1.x 的读者会发现，与 Struts 1 相比，Struts 2 框架中的标签库发生了很大的变化。具体说明如下。

- Struts 1 框架中将标签库分为 html、bean、logic、tiles 和 nested 总共 5 个标签库，在使用时需要在页面中分别导入；Struts 2 框架的标签库不进行分类，只有一个标签库，简化了标签库的导入。
- Struts 1.X 标签并不支持表达式语言，如果需要在 Struts 1.X 中使用表达式语言，则需要安装相应的插件；Struts 2 框架默认支持 OGNL、JSTL、Groovy 和 Velocity 等表达式。
- Struts 1.X 标签库只支持 JSP 等少数表现层技术，但是 Struts 2 标签库可以支持多种表现层技术，例如，JSP、FreeMarker 和 Velocity 等。

16.4.3　Struts 2 标签的使用

在具体 Java Web 开发过程中，主要通过两个步骤来使用 Struts 2 标签库，分别是开发标签库实现类和定义标签库定义文件（TLD 文件）。在 Struts 2 框架中，提供了标签的处理实现

类和标签库定义文件，读者可以从“lib”文件夹中找到 struts2-core2.1.6.jar 文件，解压缩后找到 struts-tags.tld 文件，这样就可以查看 Struts 2 标签的使用方法，如图 16-10 所示。

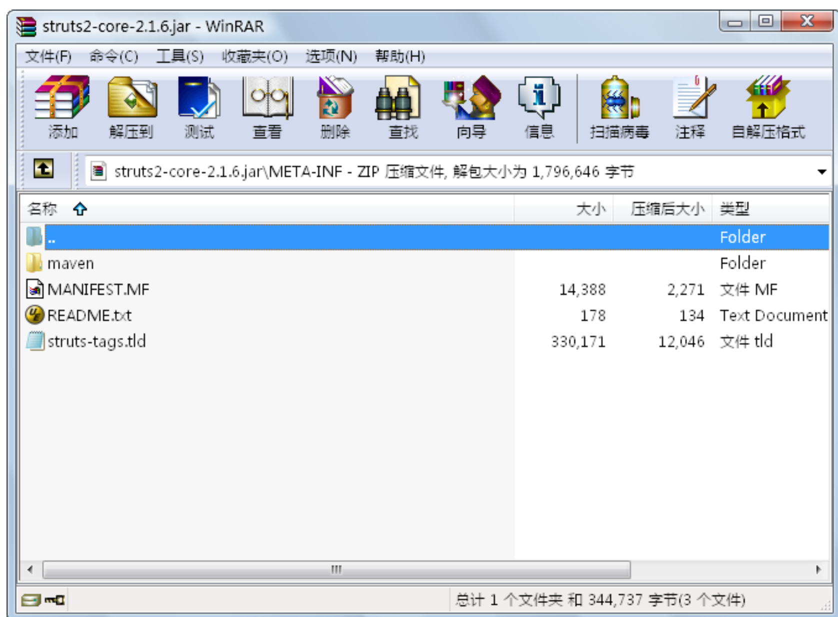


图 16-10　Struts 2 的标签库定义文件

文件 struts-tags.tld 是一个标准的 XML 文件，截取里面的一段代码如下。

```
        <?xml version="1.0" encoding="UTF-8"?>
        <taglib xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.
w3.org/2001/XMLSchema-instance"version="2.0"xsi:schemaLocation="http://java.
sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-jsptaglibrary_2_0.xsd">
          <description><![CDATA["To make it easier to access dynamic data;
                          the Apache Struts framework includes a library of
custom tags.
                          The tags interact with the framework's validation and
internationalization features;
                          to ensure that input is correct and output is
localized.
                          The Struts Tags can be used with JSP FreeMarker or
Velocity."]]></description>
          <display-name>"Struts Tags"</display-name>
          <tlib-version>2.2.3</tlib-version>
          <short-name>s</short-name>
          <uri>/struts-tags</uri>
          <tag>
            <description><![CDATA[Execute an action from within a view]]>
</description>
            <name>action</name>
            <tag-class>org.apache.struts2.views.jsp.ActionTag</tag-class>
```

```
            <body-content>JSP</body-content>
            <attribute>
              <description><![CDATA[Whether the result of this action (probably
a view) should be executed/rendered]]></description>
              <name>executeResult</name>
              <required>false</required>
              <rtexprvalue>false</rtexprvalue>
            </attribute>
            <attribute>
              <description><![CDATA[Whether the writer should be flush upon end
of action component tag, default to true]]></description>
              <name>flush</name>
              <required>false</required>
              <rtexprvalue>false</rtexprvalue>
            </attribute>
            <attribute>
              <description><![CDATA[Deprecated. Use 'var' instead]]></description>
              <name>id</name>
              <required>false</required>
              <rtexprvalue>false</rtexprvalue>
            </attribute>
            <attribute>
              <description><![CDATA[Whether  the  request  parameters  are  to  be
included when the action is invoked]]></description>
              <name>ignoreContextParams</name>
              <required>false</required>
              <rtexprvalue>false</rtexprvalue>
            </attribute>
            <attribute>
              <description><![CDATA[Name of the action to be executed (without
the extension suffix eg. .action)]]></description>
              <name>name</name>
              <required>true</required>
              <rtexprvalue>false</rtexprvalue>
            </attribute>
            <attribute>
```

倘若需要在 JSP 页面中使用 Struts 2 标签库，则需要使用<taglib-location…/>元素来指定标签库定义文件（TLD）的位置。标准使用 Struts 2 标签库的格式如下。

```
        <%@ page contentType="text/html;charset=UTF-8" language="java"%>
        <--导入 Struts 2 标签库-->
        <%@ taglib prefix="s" uri="/struts-tags"%>
```

在上述代码中，“s”表示标签库默认的短名，可以在 JSP 的标签使用中引用。具体如何使用，在前面的 JSP 中已经讲解，这里不再赘述。

提示：如果开发者使用 Servlet 2.3 或者更早的规范时，则需要在文件 web.xml 中定义标签库后，才能够在页面中使用 TagLib 导入标签库。

16.4.4 表单标签

在本书 16.1 节中曾经定义了一个表单，其实在 Struts 2 里还有另外一种写法来实现。通过这种写法可以让读者更能明白 Struts 2 的作用。接下来开始尝试改变 16.1 节中的登录页面，在此页面中尝试使用 Struts 2 标签。具体代码如下。

```
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
<head>
 <title>Login</title>
</head>
<body>
<s:form action="Logon">
  <s:textfield label="User Name" name="username"/>
 <s:password label="Password" name="password" />
 <s:submit/>
</s:form>
</body>
</html>
```

与 16.1 节中的功能对比，最重要的代码如下。

```
<%@ page language="java"import="java.util.*"pageEncoding="ISO-8859-1"%>
<%
String path = request.getContextPath();
String basePath = request.getScheme()+"://"+request.getServerName()+":
"+request.getServerPort()+path+"/";
%>
<html>
  <head>
    <base href="<%=basePath%>">
        <title>My JSP 'login.jsp' starting page</title>
          <meta http-equiv="pragma" content="no-cache">
   <meta http-equiv="cache-control" content="no-cache">
   <meta http-equiv="expires" content="0">
   <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
   <meta http-equiv="description" content="This is my page">
   <!--
   <link rel="stylesheet" type="text/css" href="styles.css">
   -->
  </head>
    <body>
  <form action="login.action" method="post">
  username:<input type="text" name="username"><br>
```

```
        password:<input type="password" name="password"><br>
        <input type="submit" value="submit">
        </form>

        </body>
    </html>
```

执行上述代码后会发现和 16.1 节中代码的执行效果是一样的，如图 16-11 所示。

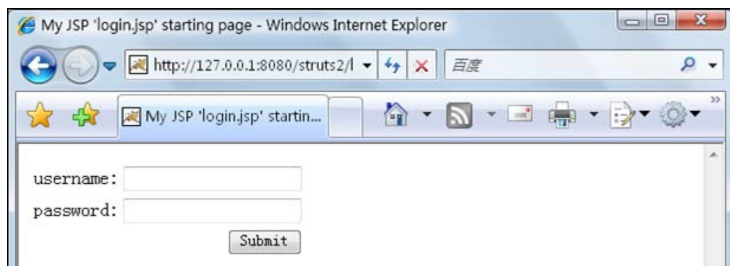


图 16-11　表单中的标签

如果仔细观看代码，会发现表单标签在仿写 HTML 代码，代码十分简单。

16.4.5　控制标签

在 Struts 2 标签库中，使用控制标签可以实现流程控制。在 Struts 2 标签中的控制标签很少，但是控制流程够用。在 Struts 2 中主要有以下 9 个控制标签。

- if：用于选择输出的标签。
- if / elseif 标签："if"标签可单独使用，也可以和'else if'标签和（或）一个多个'else'一起使用。
- else：与 if 标签结合使用，用户选择控制输出的标签。
- append：用于将多个集合拼接成一个新的集合。
- generator：一个字符串解析器，用于将一个字符串解析成一个集合。
- iterator：一个迭代器，用于将集合迭代输出。
- merge：用于将多个集合拼接成一个集合，与 append 有所不同。
- sort：用于对集合进行排序。
- subset：用于截取集合的部分元素。

实例 34：使用控制标签

本实例的目的是讲解控制标签的用法，演示了 if 标签的使用流程。本实例的实现代码如下。

```
    <s:if test="%{false}">
        <div>Will Not Be Executed</div>
    </s:if>
    <s:elseif test="%{true}">
        <div>Will Be Executed</div>
    </s:elseif>
```

```
    <s:else>
        <div>Will Not Be Executed</div>
    </s:else>
```

这段程序没有任何运行结果，目的是让读者明白 if 标签的使用流程。其实就和前面讲解的 if 语句差不多，只是写的方式有所不同。


多学一招

这些标签的使用方法都十分简单，在互联网上有许多相关的知识，读者要想全部掌握它，需要查阅相关资料，但是想融会贯通，还得下一番功夫。

16.4.6 Struts 2 常用标签

Struts 2 里的标签没有分类，在 JSP 头文件中加上<%@taglib prefix="s" uri="/struts-tags" %>后就能使用 Struts 2 中的标签库。在 Struts 2 中的常用标签信息如下。

- <s:a href=""></s:a>：超链接，类似于 HTML 里的<a>。
- <s:action name=""></s:action>：执行一个 view 里面的一个 action。
- <s:actionerror/>：如果 action 的 errors 有值那么显示出来。
- <s:actionmessage/>：如果 action 的 message 有值那么显示出来。
- <s:append></s:append>−：添加一个值到 list，类似于 list.add()。
- <s:autocompleter></s:autocompleter>：自动完成<s:combobox>标签的内容，即 Ajax。
- <s:bean name=""></s:bean>：类似于 struts1.x 中的 JavaBean 的值。
- <s:checkbox></s:checkbox>：复选框。
- <s:checkboxlist list=""></s:checkboxlist>：多选框。
- <s:combobox list=""></s:combobox>：下拉框。
- <s:component></s:component>：图像符号。
- <s:date/>：获取日期格式。
- <s:datetimepicker></s:datetimepicker>：日期输入框。
- <s:debug></s:debug>：显示错误信息。
- <s:div></s:div>：表示一个块，类似于 HTML 的<div></div>。
- <s:doubleselect list="" doubleName="" doubleList=""></s:doubleselect>：双下拉框。
- <s:if test=""></s:if>、<s:elseif test=""></s:elseif>、<s:else></s:else>：这 3 个标签一起使用，表示条件判断。
- <s:fielderror></s:fielderror>：显示文件错误信息。
- <s:file></s:file>：文件上传。
- <s:form action=""></s:form>：取相应 form 的值。
- <s:generator separator="" val="">、</s:generator>：和<s:iterator>标签一起使用。
- <s:head/>：在<head></head>里使用，表示头文件结束。
- <s:hidden></s:hidden>：隐藏值。
- <s:i18n name=""></s:i18n>：加载资源包到值堆栈。
- <s:include value=""></s:include>：包含一个输出，servlet 或 jsp 页面。

- <s:inputtransferselect list=""></s:inputtransferselect>：获取 form 的一个输入。
- <s:iterator></s:iterator>：用于遍历集合。
- <s:label></s:label>：只读的标签。
- <s:merge></s:merge>：合并遍历集合出来的值。
- <s:optgroup></s:optgroup>：获取标签组。
- <s:optiontransferselect doubleList="" list="" doubleName=""></s:optiontransferselect>：左右选择框。
- <s:param></s:param>：为其他标签提供参数。
- <s:password></s:password>：密码输入框。
- <s:property/>：得到‘value’的属性。
- <s:push value=""></s:push>：value 的值 push 到栈中，从而使 property 标签能够获取 value 的属性。
- <s:radio list=""></s:radio>：单选按钮。
- <s:reset></s:reset>-：重置按钮。
- <s:select list=""></s:select>：单选框。
- <s:set name=""></s:set>：赋予变量一个特定范围内的值。
- <s:sort comparator=""></s:sort>：通过属性给 list 分类。
- <s:submit></s:submit>：提交按钮。
- <s:subset></s:subset>：为遍历集合输出子集。
- <s:tabbedPanel id=""></s:tabbedPanel>：表格框。
- <s:table></s:table>：表格。
- <s:text name=""></s:text>：显示文本信息。
- <s:textarea></s:textarea>：文本域输入框。
- <s:textfield></s:textfield>：文本输入框。
- <s:token></s:token>：拦截器。
- <s:tree></s:tree>：树。
- <s:treenode label=""></s:treenode>：树的结构。
- <s:updownselect list=""></s:updownselect>：多选择框。
- <s:url></s:url>：创建 url。

16.5　编写 Action

Action 是 Struts 2 的核心，在开发 Java Web 项目时，开发者需要编写大量的 Action 类，并且需要在文件 Struts.xml 中配置 Action。在 Action 类中包含了对用户请求的处理逻辑，Action 类也被称为业务控制器。例如，可以通过如下代码处理一个获取计算机时间的 Action。

```
import java.text.DateFormat;
import java.util.Date;
import com.opensymphony.xwork2.ActionSupport;
```

```
            public class HelloWorld extends ActionSupport
        {
             private String message;
                 public String getMessage()
        {
                 return message;
            }
                @Override
              public String execute()
        {
                message = " Hello World, Now is " + DateFormat.getInstance().
format ( new Date());
                 return SUCCESS;
            }
```

16.5.1 使用 Action 动态调用

动态调用 Action 是十分常见的，例如，在一个表单中，当用户通过不同的按钮提交一个表单时，系统将会调用不同的 Action 方法去处理信息，此时就需要 Action 实现多个控制处理逻辑。在 Struts 2 中，有多个处理逻辑的 Action 的方法，下面通过几段代码（光盘：源代码/第 16 章/16-3）进行讲解，具体实现流程如下。

1）部署程序环境，将如图 16-12 所示的 Jar 文件复制到项目中去。

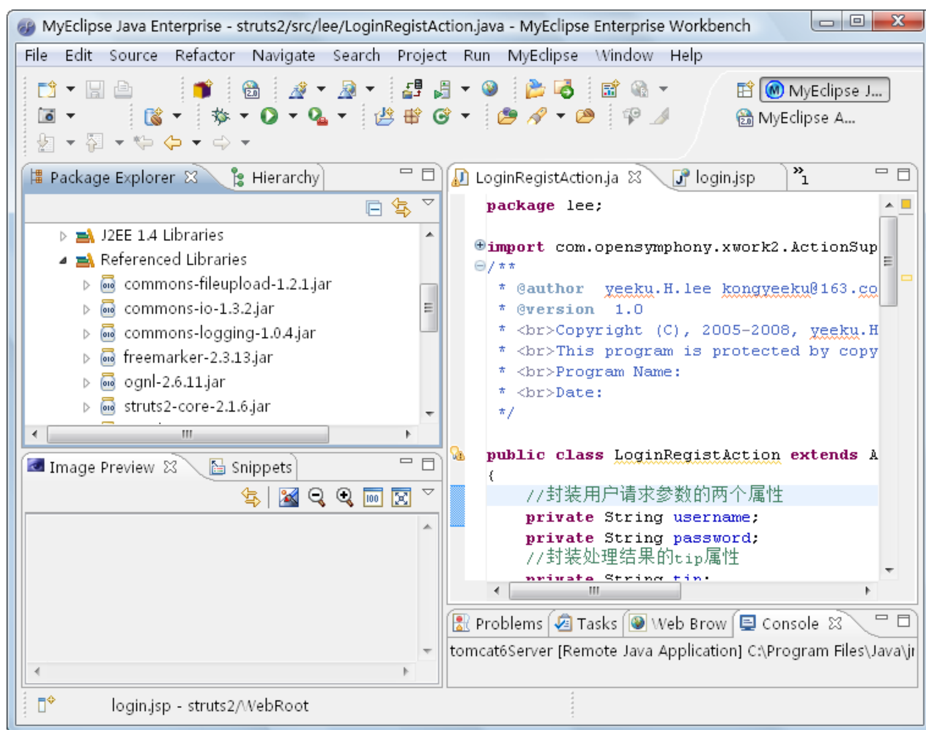


图 16-12 复制 Struts 2 配置文件

2）修改文件 web.xml，修改后的代码如下。

```
<?xml version="1.0" encoding="GBK"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
   http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
   <!-- 定义 Struts2 的核心 Filter -->
   <filter>
        <filter-name>struts2</filter-name>
        <filter-class>org.apache.struts2.dispatcher.FilterDispatcher
</filter-class>
   </filter>
   <!-- 让 Struts2 的核心 Filter 拦截所有请求 -->
   <filter-mapping>
        <filter-name>struts2</filter-name>
        <url-pattern>/*</url-pattern>
   </filter-mapping>
</web-app>
```

3）在文件 web.xml 的同目录下新建一个 bulid.xml 文件，其代码如下。

```
<?xml version="1.0"?>
<project name="struts" basedir="." default="">
    <path id="classpath">
        <fileset dir="lib">
            <include name="*.jar"/>
        </fileset>
        <pathelement path="."/>
    </path>
   <target name="compile" description="Compile all source code">
         <delete dir="classes"/>
         <mkdir dir="classes"/>
         <copy todir="classes">
           <fileset dir="src">
                    <exclude name="**/*.java"/>
                </fileset>
         </copy>
        <javac destdir="classes" debug="true"
            deprecation="false" optimize="false" failonerror="true">
            <src path="src"/>
            <classpath refid="classpath"/>
        </javac>
    </target>
</project>
```

4）新建一个 lee 包，然后新建文件 LoginRegistAction.java，其源代码如下。

```
package lee;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ActionContext;
public class LoginRegistAction extends ActionSupport
{
   //封装用户请求参数的两个属性
   private String username;
   private String password;
   //封装处理结果的 tip 属性
   private String tip;
   //username 属性对应的 setter 和 getter 方法
   public String getUsername()
   {
        return username;
   }
   public void setUsername(String username)
   {
        this.username = username;
   }
   //password 属性对应的 getter 和 setter 方法
   public String getPassword()
   {
        return password;
   }
   public void setPassword(String password)
   {
        this.password = password;
   }
   //tip 属性对应的 setter 和 getter 方法
   public String getTip()
   {
        return tip;
   }
   public void setTip(String tip)
   {
        this.tip = tip;
   }
   //Action 包含的注册控制逻辑
   public String regist() throws Exception
   {
       ActionContext.getContext().getSession().put("user" , getUsername());
        setTip("恭喜您," + getUsername() + ",您已经注册成功！");
        return SUCCESS;
   }
   //Action 默认包含的控制逻辑
   public String execute() throws Exception
   {
        if (getUsername().equals("scott")
```

```
                      && getPassword().equals("tiger") )
                {
                      ActionContext.getContext().getSession().put("user",getUsername());
                      setTip("欢迎," + getUsername() + ",您已经登录成功! ");
                      return SUCCESS;
                }
                else
                {
                      return ERROR;
                }
          }
     }
```

5）在“str”目录下新建文件 struts.xml，其代码如下。

```
<?xml version="1.0" encoding="GBK"?>
<!DOCTYPE struts PUBLIC
      "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
      "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>

   <package name="lee" extends="struts-default">
      <action name="login" class="lee.LoginRegistAction">
         <result name="input">/login.jsp</result>
         <result name="error">/error.jsp</result>
         <result name="success">/welcome.jsp</result>
      </action>
   </package>
</struts>
```

6）启动 Tomcat 进行浏览，运行后得到如图 16-13 所示的页面。

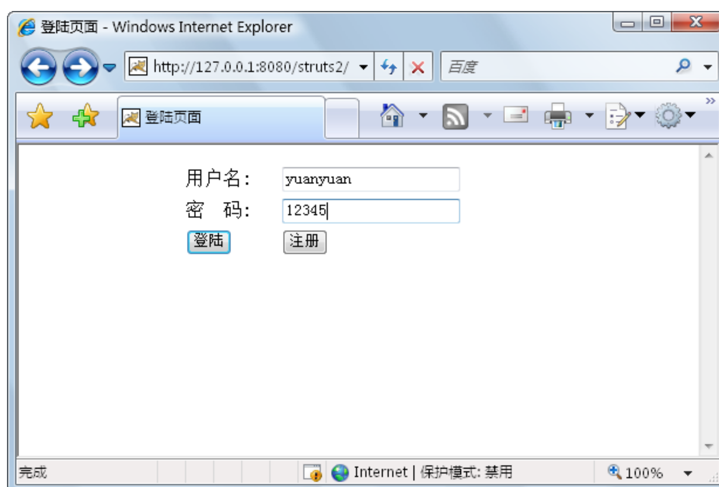


图 16-13　浏览首页

在表单中输入“用户名”和“密码”信息，单击“注册”按钮后可以看到如图 16-14 所示的效果。



图 16-14　动态方法调用

上面这种方式，可以在一个 Action 中调用多个逻辑处理，并通过为表单元素指定不同的 Action 属性来提交不同的 Action 方法。

16.5.2　使用 Action 进行跳转

跳转在 Struts 2 中是一个十分普遍的动作，在 Struts 2 中完成跳转是请求资源的一种方法。下面将通过一个简单的例子来讲解使用 Action 进行跳转的流程，读者可以根据这段代码明白请求资源是怎么一回事。要开发这样的程序，首先要部署如图 16-15 所示的 jar 文件，代码保存在（光盘：源代码/第 16 章/16-4）中，具体实现流程如下。

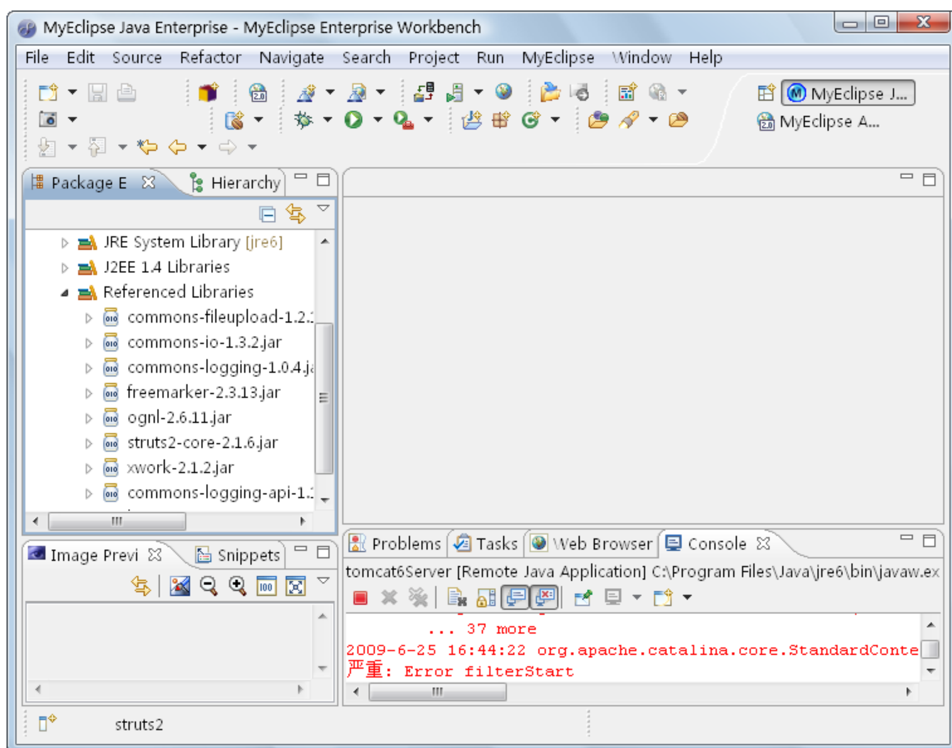


图 16-15　部署 struts 环境

1）处理请求，用一个属性来封装请求参数，并提供一个参数来封装处理后的提示，新建文件 MyAction.java 来实现此功能，具体代码如下。

```
package lee;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ActionContext;
public class MyAction extends ActionSupport
{
   //封装请求参数的 target 属性。
   private String target;
   //封装处理结果的 tip 属性
   private String tip;
   //此处省略了 target 和 tip 属性的 setter 和 getter 方法
   public void setTarget(String target)
   {
        this.target = target;
   }
   public String getTarget()
   {
        return this.target;
   }

   public void setTip(String tip)
   {
        this.tip = tip;
   }
   public String getTip()
   {
        return this.tip;
   }

   public String execute() throws Exception
   {
        setTip("恭喜您,您已经成功转向");
        return SUCCESS;
   }
}
```

2）新建文件 login.jsp，其页面代码如下。

```
<%@ page language="java" contentType="text/html; charset=GBK"%>
<html>
<head>
<title>转向页面</title>
</head>
<body>
<table width="400" align="center">
```

```
        <form action="MyAction.action" method="post">
        <tr>
        <td>转入的目标页面:</td>
        <td><input type="text" name="target"/></td>
        </tr>
        <tr>
        <td colspan="2">注意：<br>您应该输入welcome，系统只提供welcome.jsp 页面</td>
        </tr>
        <tr>
        <td colspan="2"><input type="submit" value="转入"/></td>
        </tr>
        </form>
        <table>
        </body>
        </html>
```

3）因为这是首页，要想进行跳转就必须新建一个需要跳转的页面。新建文件 welcome.jsp，其页面代码如下。

```
        <%@ page language="java" contentType="text/html; charset=GBK"%>
        <%@taglib prefix="s" uri="/struts-tags"%>
        <html>
           <head>
                <title>成功页面</title>
           </head>
           <body>
                <s:property value="tip"/>
           </body>
        </html>
```

4）建立文件后，修改文件 web.xml，修改后的代码如下。

```
        <?xml version="1.0" encoding="GBK"?>
        <web-app xmlns="http://java.sun.com/xml/ns/javaee"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
           http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
           <!-- 定义Struts2 的核心 Filter -->
           <filter>
                <filter-name>struts2</filter-name>
                <filter-class>org.apache.struts2.dispatcher.FilterDispatcher
</filter-class>
           </filter>
           <!-- 让 Struts2 的核心 Filter 拦截所有请求 -->
           <filter-mapping>
                <filter-name>struts2</filter-name>
                <url-pattern>/*</url-pattern>
```

```
    </filter-mapping>
</web-app>
```

5）新建一个文件 bulid.xml，其代码如下。

```
<?xml version="1.0"?>
<project name="struts" basedir="." default="">
    <path id="classpath">
        <fileset dir="lib">
            <include name="*.jar"/>
        </fileset>
        <pathelement path="."/>
    </path>
   <target name="compile" description="Compile all source code">
          <delete dir="classes"/>
          <mkdir dir="classes"/>
          <copy todir="classes">
            <fileset dir="src">
                     <exclude name="**/*.java"/>
                </fileset>
          </copy>
        <javac destdir="classes" debug="true"
            deprecation="false" optimize="false" failonerror="true">
            <src path="src"/>
            <classpath refid="classpath"/>
        </javac>
    </target>
</project>
```

6）编写文件 struts.xml，其代码如下。

```
<?xml version="1.0" encoding="GBK"?>
<!DOCTYPE struts PUBLIC
   "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
   "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
   <package name="lee" extends="struts-default">
         <!-- 配置处理用户请求的 Action -->
         <action name="MyAction" class="lee.MyAction">
              <!-- 配置 Result，使用 OGNL 表达式来指定视图资源 -->
              <result name="success">/${target}.jsp</result>
         </action>
         <action name="">
              <result>.</result>
         </action>
   </package>
</struts>
```

启动 Tomcat 并输入网址，运行后会得到如图 16-16 所示的效果，转向后会打开如图 16-17 所示的效果。

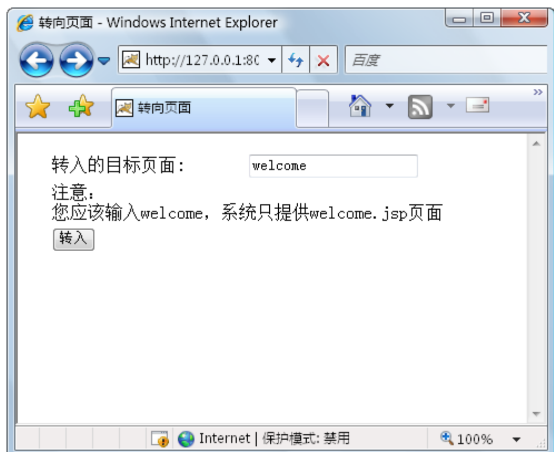


图 16-16　得到的结果

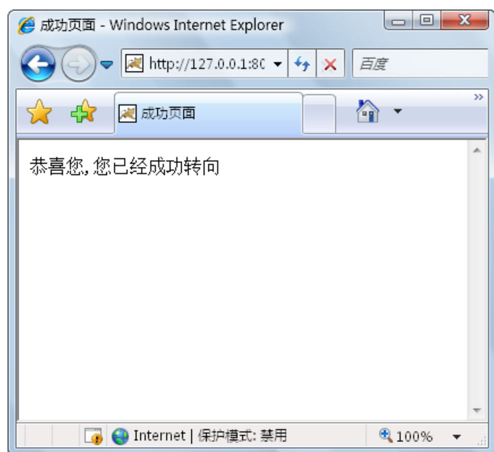


图 16-17　输入正确后

如果输入的不是这个关键字，将会打开如图 16-18 所示的效果。

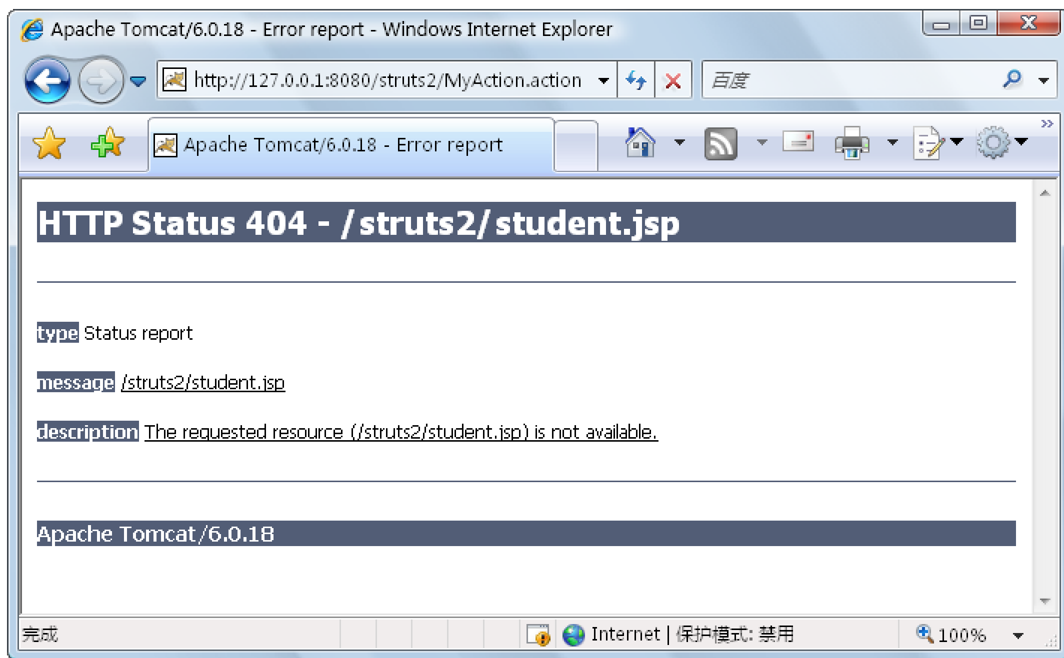


图 16-18　找不到这个页面

16.5.3　编写 Action 的经典法则

在 Web 开发过程中，开发人员经常忽视展现层 Action 的单元测试，主要有如下两条原因。

1）Action 层的业务逻辑比较简单，开发人员会认为这一部分代码不重要。

2）Action 层难以模拟 http 请求传递参数，需要依赖 web 容器，因此给单元测试编写带来一定的难度。

基于 Struts 2 的 mock 和 WebWork 的 ActionProxyFactory，都可以进行 Action 的单元测试。大多数程序员倾向于用 ActionProxyFactory 做单元测试。写 Action 的过程通过如下 3 步完成。

1）设置 ActionContext 上下文参数，将表单传递的请求参数添加到 map 中。

2）创建 Action 动态代理对象，通过 public abstract ActionProxy createActionProxy(String namespace, String actionName, Map extraContext) throws Exception 创建 Action 代理对象。

3）用 Junit 断言执行结果，方法是 assertEquals(testAction.login(),"success")。

代码如下。

```
public class TestActionTest extends BaseCaseTest{
private ActionProxy proxy = null;
private IVoucherService voucherService;
@Before
public void setUp() throws Exception{
IMocksControl control = EasyMock.createControl();
voucherService = control.createMock(IVoucherService.class);
Map<String, Object> params = new HashMap<String, Object>();
params.put("loginId","test");
params.put("password","111111");
params.put("voucherService", voucherService);
Map extraContext = new HashMap();
extraContext.put(ActionContext.PARAMETERS,params);
try
{
proxy   =   ActionProxyFactory.getFactory().createActionProxy("/user",
"testAction", extraContext);
proxy.setExecuteResult(false);
assertEquals(proxy.execute(),"success");
} catch (Exception e) {
e.printStackTrace();
}
}
public void testLogin()
 {
TestAction testAction = (TestAction) proxy.getAction();
assertEquals(testAction.login(),"success");
}
}
```

在创建代理 Action 时，一定要执行 proxy.execute()方法，否则参数不能够增加到 ActionContext 上下文中。因为在 proxy.execute()中会执行 invocation.invoke()的核心方法，所以需要遍历执行 Action 中所有的拦截器，包括其中的参数拦截器。

16.6 疑难问题解析

本章详细讲解了 Struts 2 的基础知识。本节将对本章中比较难理解的问题进行讲解。

读者疑问：怎样建立一个适合 Struts 2 工程的开发与运行环境呢？我配置很久都感觉开发不成功，其原因在什么地方？

解答：用户首先安装 JDK，然后安装 Tomcat，用户还需要安装 MyEclipse，最关键的一步是，用户需要在 Tomcat 里打开 server.xml，添加一个虚拟目录，虚拟目录用户一定要与工程的目录相同，要指到 WebRoot 目录下，然后将 Tomcat 加载到 MyEclipse 里，这些知识前面都讲过，如果读者不会，可以去光盘中查看相关的视频，学会详细配置 Struts 2 的开发环境。

读者疑问：Struts 环境配置正确了，可是调试 web 程序，总是无法显示网页，这是为什么呢？

解答：原因很多，用户首先要检查 Java 的执行环境是不是正确，如果正确，用户需要检查 Tomcat 和 MyEclipse 是不是配置正确，如果配置正确，用户查看是不是需要的 Jar 已经加载进去，整个过程十分烦琐，用户需要十分细心，从 Java 运行环境检查起，然后逐一排除，一定可以找出问题所在。



职场点拨——谈学习效率

学习软件开发是充满挑战之路，也是充满乐趣之路。学习 Java Web 开发也是如此，没有捷径可走。笔者在此提出如下 3 条建议来作为读者快速学习编程的技巧。

（1）培养兴趣

兴趣是能够让你坚持下去的动力。无论做什么事情，只要有了兴趣，你就喜欢花费时间去做它。只要你喜欢感受那调试成功的喜悦，就说明你已经对编程产生了兴趣。另外调试成功的喜悦会让你更加喜欢编程，更能给你带来成就感。有时间多到一些程序员论坛转一转，你会发现，他们其实很乐观幽默，并且能够帮你解决很多问题。

（2）脚踏实地

都说欲速则不达，学编程切忌有浮躁的心态。很多初学者刚学会了基本语法知识，调试成功了几段代码，就迫不及待大声宣布“我精通 Java 了”。但是当遇到问题之后才发现，自己学到的只是九牛一毛。

（3）多实践，要自己动手

有前辈高人认为，学习编程的秘诀是：编程、编程、再编程，练习、练习、再练习。笔者深表赞同。软件开发很强调实践动手能力，所以实践就变得尤为重要。学编程不仅要多实践，而且要快实践。在看书的时候，不要等到完全理解了才动手，而是应该在看书的同时输入代码，程序运行的各种情况可以让你更快、更牢固地掌握知识点。

第 17 章　Struts 2 架构

通过第 16 章的学习，相信读者已经明白了 Struts 2 框架的模块化、灵活性和重用性的优点。在本章将详细讲解 Struts 2 架构的知识，将会了解 Struts 2 是如何清晰地区分控制、事务逻辑和外观的，从而简化了开发应用程序过程。通过本章内容，将向读者讲解拦截器、类型转换器、校验支持、Struts 对 Ajax 的支持和文件上传、下载的知识。通过本章能学到如下知识。

- 拦截器。
- 类型转换器。
- 文件的上传和下载。
- 职场点拨——教会徒弟饿死师傅。

2011 年 XX 月 XX 日，小雪

今天开早会，项目经理让我和 HR 合作，招收一个新人。老大嘱咐我向 HR 学习相关的招聘知识，以便尽快找到合适的人选。可是 HR 总是对我爱理不理，此事也就一直没有进展。



一问一答

小菜：“好烦啊，HR 究竟是什么意思呢？”

Wisdom：“在公司内部，让有经验者指导新人的案例常见，因为领导都想让员工们全面发展，共同为企业创造价值。但是老员工会心甘情愿吗？当然不是，俗话说教会徒弟，饿死师傅。”

小菜：“哎，还是学习吧！其实我有一个问题：已经有了 Struts 1，还要 Struts 2 有什么用呢？”

Wisdom：“新的技术都会借鉴旧技术的精华，从而将旧的取代。Struts 2 虽然不是 Struts 1 的完全取代，但是今日你在哪儿能找到 Struts 1？只会见到 Struts 2 在这里呼风唤雨。”

17.1　认识全新的 Struts 2

新的技术都会借鉴旧技术的精华，从而取而代之。Struts 2 虽然不能完全取代 Struts 1，但是如今很难找到 Struts 1 的应用，只见到 Struts 2 在呼风唤雨。这个框架之所以叫

“Struts”，是为了提醒我们记住那些支撑我们的房屋、建筑、桥梁，甚至我们踩高楼时候的基础支撑。这也是一个解释 Struts 在开发 Web 应用程序中所扮演的角色的精彩描述。本章主要讲解 Struts 2 的架构知识。下面通过一段代码来演示 Struts 2 的功能，实现代码保存在（光盘：源代码/第 17 章/kai/）中，具体实现流程如下。

1）新建文件 login.jsp，其代码如下。

```
        <%@ page language="java" contentType="text/html; charset=GBK"%>
        <%@taglib prefix="s" uri="/struts-tags"%>
        <html>
        <head>
        <title>登录页面</title>
        </head>
        <body>
        <div style="color:red" align="center">${requestScope.tip}<div>
        <form action="login.action" method="post">
            <table align="center">
            <caption><h3>用户登录</h3></caption>
                <tr>
                    <td>用户名：<input type="text" name="username"/></td>
                </tr>
                <tr>
                    <td>密  码：<input type="text"name="password"/></td>
                </tr>
                <tr align="center">
                    <td><input  type="submit"  value="登录"/><input  type="reset"
value="重填" /></td>
                </tr>
            </table>
        </form>

        </body>
        </html>
```

2）新建文件 welcome.jsp，具体代码如下。

```
        <%@ page language="java" contentType="text/html; charset=GBK"%>
        <html>
            <head>
                <title>成功页面</title>
            </head>
            <body>
                您已经登录!<br>
           <a href="viewBook.jsp">查看购物车</a>
            </body>
        </html>
```

3）新建处理登录失败的页面文件 error.jsp，其实现代码如下。

```
<%@ page language="java" contentType="text/html; charset=GBK"%>
<html>
    <head>
        <title>错误页面</title>
    </head>
    <body>
        您不能登录!<br>
    </body>
</html>
```

4）新建一个 AuthorityInterceptor 类，其代码如下。

```
package lee;
import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.Action;
import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.interceptor.AbstractInterceptor;
import java.util.*;
//权限检查拦截器继承类 AbstractInterceptor
public class AuthorityInterceptor extends AbstractInterceptor
{
   //拦截 Action 处理的拦截方法
    public String intercept(ActionInvocation invocation)
         throws Exception
   {
         //取得请求相关的 ActionContext 实例
         ActionContext ctx = invocation.getInvocationContext();
         Map session = ctx.getSession();
         //取出名为 user 的 Session 属性
         String user = (String)session.get("user");
         //如果没有登录，或者登录所用的用户名不是 scott，则都返回重新登录
         if (user != null && user.equals("Login.jsp") )
         {
              return invocation.invoke();
         }
         //没有登录，将服务器提示设置成一个 HttpServletRequest 属性
         ctx.put("tip" , "您还没有登录，请输入你的用户名和密码");
         //直接返回 login 的逻辑视图
         return Action.LOGIN;
    }
}
```

5）新建一个 LoginAction 类，其代码如下。

```
package lee;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ActionContext;
import java.util.*;
public class LoginAction extends ActionSupport
{
   private String username;
   private String password;
   public void setUsername(String username)
   {
      this.username = username;
   }
   public String getUsername()
   {
      return username;
   }
   public void setPassword(String password)
   {
      this.password = password;
   }
   public String getPassword()
   {
      return password;
   }
   public String execute() throws Exception
   {
       System.out.println("进入 execute 方法执行体.........");
       Thread.sleep(1500);
      if (getUsername().equals("admin")
            && getPassword().equals("admin") )
       {
            ActionContext ctx = ActionContext.getContext();
            Map session = ctx.getSession();
            session.put("user" , getUsername());
         return SUCCESS;
      }
       else
       {
         return ERROR;
      }
   }
}
```

6）实现配置文件 struts.xml，其代码如下。

```
<?xml version="1.0" encoding="GBK"?>
<!DOCTYPE struts PUBLIC
```

```
    "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
    <constant name="struts.custom.i18n.resources" value="globalMessages"/>
    <constant name="struts.i18n.encoding" value="GBK"/>
    <package name="lee" extends="struts-default">
        <!-- 用户拦截器定义在该元素下 -->
        <interceptors>
            <!-- 定义了一个名为 authority 的拦截器 -->
            <interceptor name="authority" class="lee.AuthorityInterceptor"/>
        </interceptors>
        <!-- 定义全局 Result -->
        <global-results>
            <!-- 当返回 login 视图名时，转入/login.jsp 页面 -->
            <result name="login">/login.jsp</result>
        </global-results>
        <action name="login" class="lee.LoginAction">
            <result name="error">/error.jsp</result>
            <result name="success">/welcome.jsp</result>
        </action>
        <!-- 定义一个名为 viewBook 的 Action，其实现类为 ActionSupport -->
        <action name="viewBook">
            <!-- 返回 success 视图名时，转入/WEB-INF/jsp/viewBook.jsp 页面 -->
            <result>/WEB-INF/jsp/viewBook.jsp</result>
            <!-- 拦截器一般配置在 result 元素之后！ -->
            <interceptor-ref name="defaultStack"/>
            <!-- 应用自定义拦截器 -->
            <interceptor-ref name="authority"/>
        </action>
        <action name="">
            <result>.</result>
        </action>
    </package>
</struts>
```

在浏览器中输入相应的地址，然后随意输入错误的用户名和密码，会得到如图 17-1 所示的效果。



图 17-1　错误提示

当输入的用户名和密码都为 admin 时，会进入 17-2 所示的页面。

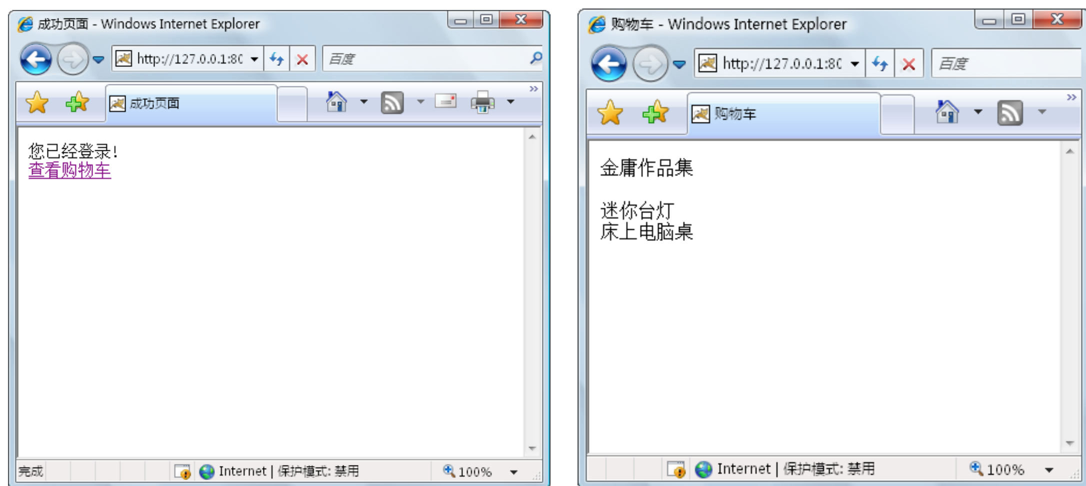


图 17-2　登录成功页面

17.2　拦截器

从 Struts 2 框架上看，拦截器完成了 70%的工作。拦截器能够解析请求参数，将请求参数赋予 Action 属性、执行数据库校验、文件上传等功能。Struts 2 的设计十分巧妙，更大程度地得益于拦截器设计，当需要扩展 Struts 2 功能时，只需要提供对应拦截器，并将它配置在 Struts 2 容器中即可。

17.2.1　什么是拦截器

拦截器是 AOP（Aspect Oriented Programming，面向方面编程）中的概念，它本身是一段代码，可以通过定义“织入点”来指定拦截器的代码在“织入点”的前后执行，从而起到拦截的作用。正如 Struts 2 的 Reference 中所描述的那样，Struts 2 的拦截对象是 Action 代码，可以在 Action 代码之前或者之后执行拦截器的代码。

拦截器在 WebWork 的中文文档中的解释如下。

拦截器是动态拦截 Action 调用的对象。它提供了一种机制，开发者可以定义在一个 Action 执行的前后执行的代码，也可以在一个 Action 执行前阻止其执行。同时也提供了一种可以提取 Action 中可重用的部分的方式。

谈到拦截器，还需要了解拦截器链。拦截器链能够将拦截器按一定的顺序联结成一条链。在访问被拦截的方法或字段时，拦截器链中的拦截器就会按其之前定义的顺序被调用。实现 Struts 2 的拦截器的方法非常简单，具体流程如下。

1）当请求到达 Struts 2 的 ServletDispatcher 时，Struts 2 会查找配置文件，并根据其配置实例化相对的拦截器对象。

2）串成一个列表（list），并一个一个地调用列表中的拦截器，如图 17-3 所示。

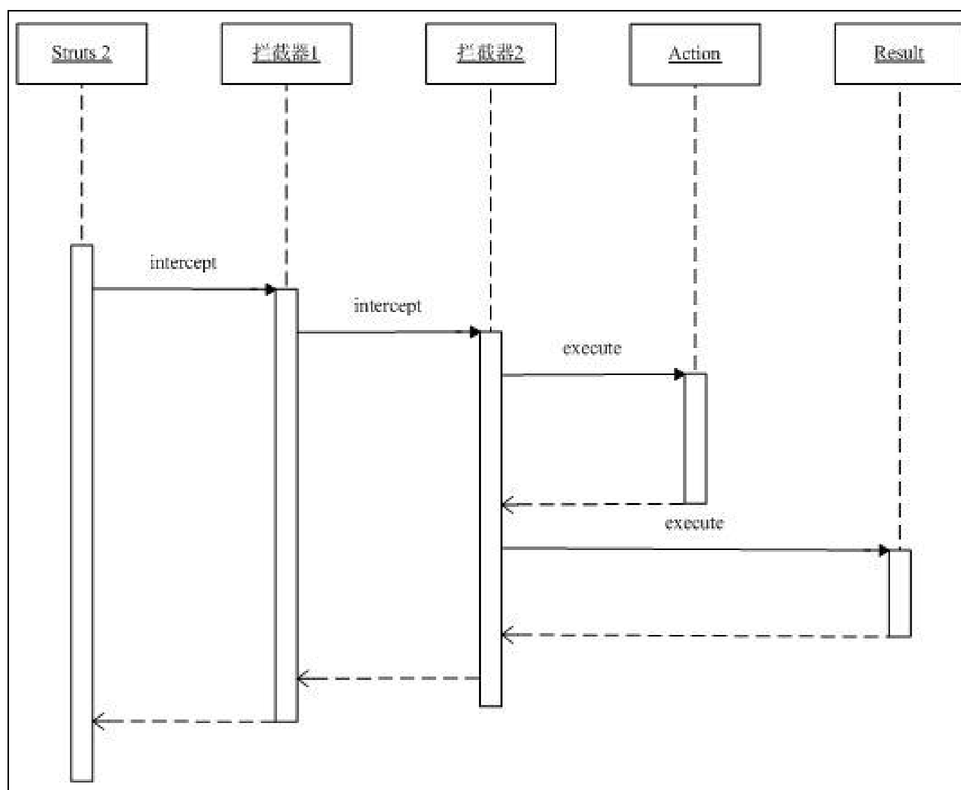


图 17-3　拦截器调用序列图

读者可以在 struts2-all-2.0.1.jar 或 struts2-core-2.0.1.jar 包的 struts-default.xml 中，查看关于默认的拦截器与拦截器链的配置知识。在本书后面的内容中，将讲解如何配置拦截器。

17.2.2　配置拦截器

在文件 struts-default.xml 中已经配置了一些拦截器，如果想要使用上述拦截器，只需要在应用程序的 struts.xml 文件中，通过“<include file="struts-default.xml" />”将 struts-default.xml 文件包含进来。打开 struts2-core-2.1.6.jar 文件，打开里面的 struts-default.xml 文件，截取一段如下所示的代码。

```
        < interceptor name ="alias" class ="com.opensymphony.xwork2.interceptor.
AliasInterceptor" />
        <interceptor name ="autowiring" class =
        "com.opensymphony.xwork2.spring.interceptor.ActionAutowiringInterceptor" />
        <interceptor   name="chain"class="com.opensymphony.xwork2.interceptor.
ChainingInterceptor" />
        <interceptor  name="conversionError"class="org.apache.struts2.interceptor.
StrutsConversionErrorInterceptor" />
        <interceptor  name="createSession"class="org.apache.struts2.interceptor.
CreateSessionInterceptor" />
```

```
        <interceptor name ="debugging" class ="org.apache.struts2.interceptor.
debugging.DebuggingInterceptor" />
        <interceptor name ="external-ref" class =
        "com.opensymphony.xwork2.interceptor.ExternalReferencesInterceptor" />
        <interceptor name="execAndWait"class ="org.apache.struts2.interceptor.
ExecuteAndWaitInterceptor" />
        <interceptor name="exception"class="com.opensymphony.xwork2.interceptor.
ExceptionMappingInterceptor" />
        <interceptor  name="fileUpload"class  ="org.apache.struts2.interceptor.
FileUploadInterceptor" />
        <interceptor name ="i18n" class ="com.opensymphony.xwork2.interceptor.
I18nInterceptor" />
        <interceptor name="logger"class ="com.opensymphony.xwork2.interceptor.
LoggingInterceptor" />
        <interceptor                                  name="model-driven"class
="com.opensymphony.xwork2.interceptor. ModelDrivenInterceptor" />
        < interceptor name ="scoped-model-driven" class =
        "com.opensymphony.xwork2.interceptor.ScopedModelDrivenInterceptor" />
        < interceptor name="params"class="com.opensymphony.xwork2.interceptor.
ParametersInterceptor" />
        < interceptor name="prepare"class="com.opensymphony.xwork2.interceptor.
PrepareInterceptor" />
        < interceptor name ="static-params" class =
        "com.opensymphony.xwork2.interceptor.StaticParametersInterceptor" />
        <  interceptor  name  ="scope"  class  ="org.apache.struts2.interceptor.
ScopeInterceptor" />
        <interceptor name="servlet-config"class="org.apache.struts2.interceptor.
ServletConfigInterceptor" />
        < interceptor name ="sessionAutowiring" class =
        "org.apache.struts2.spring.interceptor.SessionContextAutowiringInterce
ptor" />
        < interceptor name="timer"class="com.opensymphony.xwork2.interceptor.
TimerInterceptor" />
        <  interceptor  name  ="token"  class  ="org.apache.struts2.interceptor.
TokenInterceptor" />
        <interceptor name="token-session"class ="org.apache.struts2.interceptor.
TokenSessionStoreInterceptor" />
        <interceptor  name="validation"class="com.opensymphony.xwork2.validator.
ValidationInterceptor" />
        <   interceptor   name   ="workflow"   class   ="com.opensymphony.xwork2.
interceptor.DefaultWorkflowInterceptor" />
        <  interceptor  name  ="store"  class  ="org.apache.struts2.interceptor.
MessageStoreInterceptor" />
        < interceptor name ="checkbox" class ="org.apache.struts2.interceptor.
CheckboxInterceptor" />
        <  interceptor  name="profiling"class  ="org.apache.struts2.interceptor.
```

```
ProfilingActivationInterceptor" />
```

从上面的代码中可以看出，可以非常简单地定义拦截器，具体格式如下。

```
< interceptor name ="拦截器名" class ="拦截器实现类" />
```

在大部分时候，只需要通过上述格式就可以完成拦截器的配置，如果还需要用拦截器传回参数，则需要按照如下格式进行定义。

```
< interceptor name ="拦截器名" class ="拦截器实现类" />
< param name =“参数名">参数值< /param>
< / interceptor>
```

通过上述格式，在配置拦截器时，同时也传回拦截器的参数。如果将多个拦截器组合在一起组成拦截器栈，就需要配置拦截器栈，其格式如下。

```
<<interceptor-stack name="拦截器栈名">
            <interceptor-ref name="拦截器名"/>
            <interceptor-ref name="拦截器名"/>
            <interceptor-ref name="拦截器名"/>
         </interceptor-stack>
```

17.2.3　自定义拦截器

在编程过程中，用系统提供的拦截器可以完成很多操作。但有时候不够用，还需要自定义拦截器。在 Struts 2 中，自定义拦截器非常容易。在定义时要求拦截器是无状态的，这是因为 Struts 2 不能保证为每一个请求或者 Action 创建一个实例。所以如果拦截器带有状态，会引发并发问题。所有的 Struts 2 的拦截器都直接或间接实现接口 com.opensymphony.xwork2.interceptor.Interceptor。下面通过几段代码（光盘：源代码/第 17 章/17-1/）来讲解自定义拦截器的流程。

1）创建授权拦截器类 LoginAction，代码如下。

```
package lee;
import com.opensymphony.xwork2.ActionSupport;
public class LoginAction extends ActionSupport
{
   private String username;
   private String password;
   public void setUsername(String username)
  {
      this.username = username;
   }
   public String getUsername()
  {
      return username;
   }
```

```
    public void setPassword(String password)
   {
       this.password = password;
    }
    public String getPassword()
   {
       return password;
    }
    public String execute() throws Exception
   {
        System.out.println("进入execute方法执行体.........");
        Thread.sleep(1500);
       if (getUsername().equals("scott")
             && getPassword().equals("tiger") ){
          return SUCCESS;
       }
        else
        {
          return ERROR;
       }
    }}
```

2）创建授权拦截器类 SecondInterceptor，代码如下。

```
package lee;
import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.AbstractInterceptor;
import java.util.*;
public class SecondInterceptor extends AbstractInterceptor
{
   private String name;
   public void setName(String name)
   {
        this.name = name;
   }
    public String intercept(ActionInvocation invocation) throws Exception
   {
        System.out.println(name + "拦截器的动作");
        return invocation.invoke();
    }
}
```

3）创建类 SimpleInterceptor，其代码如下。

```
package lee;
import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.AbstractInterceptor;
```

```
        import java.util.*;
        public class SimpleInterceptor extends AbstractInterceptor
        {
           private String name;
           public void setName(String name)
           {
                 this.name = name;
           }
            public String intercept(ActionInvocation invocation) throws Exception
           {
                 LoginAction action = (LoginAction)invocation.getAction();
                 System.out.println(name + " 拦截器的动作---------" + "开始执行登录
Action 的时间为：" + new Date());
                 long start = System.currentTimeMillis();
                 String result = invocation.invoke();
                 System.out.println(name + " 拦截器的动作---------" + "执行完登录
Action 的时间为：" + new Date());
                 long end = System.currentTimeMillis();
                 System.out.println(name + " 拦截器的动作---------" + "执行完该
Action 的事件为" + (end - start) + "毫秒");
                 return result;
            }
        }
```

4）设置配置文件 Struts.xml，其代码如下。

```
        <?xml version="1.0" encoding="GBK"?>
        <!-- 指定 Struts2 的配置文件的 DTD 信息 -->
        <!DOCTYPE struts PUBLIC
           "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
           "http://struts.apache.org/dtds/struts-2.0.dtd">
        <struts>
           <!-- 通过常量配置 Struts2 的国际化资源信息 -->
           <constant name="struts.custom.i18n.resources" value="globalMessages"/>
           <!-- 通过常量配置 Struts2 所使用的解码集-->
           <constant name="struts.i18n.encoding" value="GBK"/>
           <!-- 配置本系统所使用的包 -->
           <package name="lee" extends="struts-default">
                 <!-- 配置应用所使用的拦截器 -->
                 <interceptors>
                       <!-- 配置 mysimple 拦截器 -->
                       <interceptor name="mysimple" class="lee.SimpleInterceptor">
                             <!-- 为拦截器指定参数值 -->
                             <param name="name">简单拦截器</param>
                       </interceptor>
                       <!-- 配置第二个拦截器 -->
                       <interceptor name="second" class="lee.SecondInterceptor"/>
```

```
                <!-- 配置名为 my-stack 的拦截器栈 -->
                <interceptor-stack name="my-stack">
                     <!-- 配置拦截器栈内的第一个拦截器 -->
                     <interceptor-ref name="mysimple">
                          <param name="name">第一个</param>
                     </interceptor-ref>
                     <!-- 配置拦截器栈内的第二个拦截器 -->
                     <interceptor-ref name="second">
                          <param name="name">第二个</param>
                     </interceptor-ref>
                </interceptor-stack>
           </interceptors>
           <!-- 配置处理用户请求的 Action -->
           <action name="login" class="lee.LoginAction">
                <!-- 配置结果映射 -->
                <result name="error">/error.jsp</result>
                <result name="success">/welcome.jsp</result>
                <!-- 拦截器一般配置在 result 元素之后！ -->
                <interceptor-ref name="defaultStack"/>
                <!-- 应用上面的拦截器栈 -->
                <interceptor-ref name="my-stack">
                     <!-- 覆盖指定拦截器的指定参数值 -->
                     <param name="second.name">改名后的拦截器</param>
                </interceptor-ref>
           </action>
           <action name="">
                <result>.</result>
           </action>
      </package>
</struts>
```

5）修改文件 web.xml，其代码和本节前面的演示代码没有多大的变化，就不再罗列。然后新建文件 login.jsp，其代码如下。

```
<%@ page language="java" contentType="text/html; charset=GBK"%>
<%@taglib prefix="s" uri="/struts-tags"%>
<html>
<head>
<title>登录页面</title>
</head>
<body>
<form action="login.action" method="post">
   <table align="center">
   <caption><h3>用户登录</h3></caption>
      <tr>
         <td>用户名：<input type="text" name="username"/></td>
      </tr>
```

```
            <tr>
                <td>密  码：<input type="text" name="password"/></td>
            </tr>
            <tr align="center">
                <td><input  type="submit"  value="登录"/><input  type="reset"
value="重填" /></td>
            </tr>
        </table>
    </form>
    </body>
    </html>
```

6）编写文件 welcome.jsp，如果登录成功会进入这个页面，其代码如下。

```
<%@ page language="java" contentType="text/html; charset=GBK"%>
<html>
    <head>
        <title>成功页面</title>
    </head>
    <body>
        您已经登录!
    </body>
</html>
```

7）编写文件 error.jsp 实现登录失败处理，其代码如下。

```
<%@ page language="java" contentType="text/html; charset=GBK"%>
<html>
    <head>
        <title>错误页面</title>
    </head>
    <body>
        您不能登录!
    </body>
</html>
```

在 MyEclipse 中启动 Tomcat，在浏览器执行时可以看到 MyEclipse 调试窗口也发生着变化，如图 17-4 所示。



如果需要使用拦截器栈时直接覆盖栈内的某个拦截器的属性值，则在指定需要被覆盖的属性时，不能只记住属性名，必须加上该属性的拦截器的名称，格式如下。

```
<拦截器名>.<属性名>
```

在浏览器中输入http://127.0.0.1:8080/struts2/Login.jsp，然后输入用户名和密码，如图 17-5 所示。

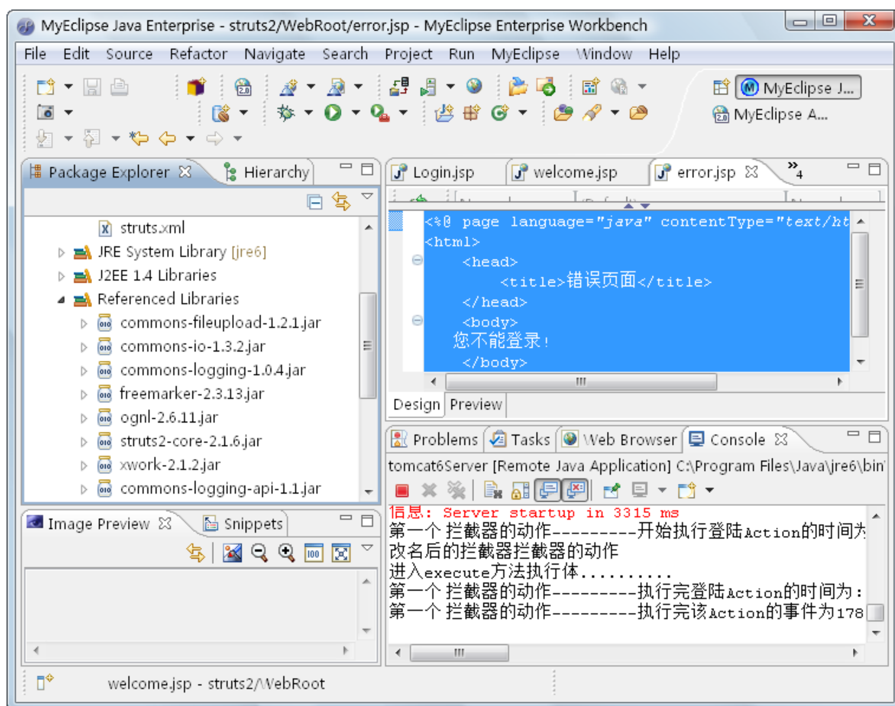


图 17-4　拦截器



图 17-5　登录页面

如果输入的用户名和密码是错误的，在 MyEclipse 窗口中可以看到不停的变化，提示拦截器的状态，而且在页面中提示用户不能登录，如图 17-6 所示。

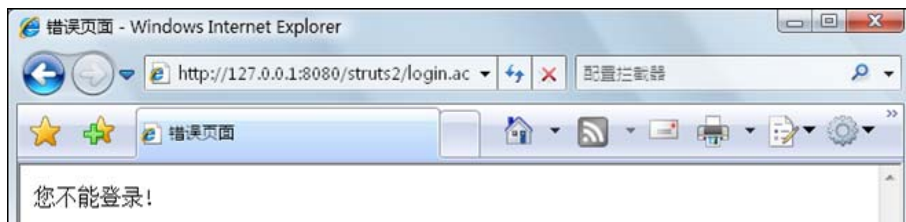


图 17-6　拦截器

17.3　类型转换器

所有的 MVC 框架都要负责解析 HTTP 请求的参数，并将参数传给控制组件。此时如果 HTTP 请求的参数都是字符类型，而 Java 是强类型的语言，所以 MVC 框架必须将这些字符串参数转换成对应的数据库。

17.3.1　OGNL 的类型转换

由于 Struts 2 支持 OGNL 表达式，所以可以使用另外一种方式将请求参数转换成复合类型。方法是在 JSP 页面中，利用 OGNL 表达式来直接生成复合类型实例。

实例 35：使用 OGNL 类型转换

本实例的目的是讲解 OGNL 类型转换的流程，其代码保存在（光盘：源代码/第 17 章/17-2/）中，具体实现流程如下。

1）新建 LoginAction 类，其代码如下。

```
package lee;
import com.opensymphony.xwork2.Action;
public class LoginAction implements Action{
   //User 类型属性封装请求参数
   private User user;
   private String tip;
   //user 属性的 setter 和 getter 方法
   public void setUser(User user)
   {
        this.user = user;
   }
   public User getUser()
   {
        return this.user;
   }
   public void setTip(String tip)
   {
        this.tip = tip;
   }
   public String getTip()
   {
        return this.tip;
   }
   public String execute() throws Exception
   {
        //通过 user 属性的 name 属性和 pass 属性来判断控制逻辑
        if (getUser().getName().equals("scott")
             && getUser().getPass().equals("tiger") )
        {
             setTip("转换成功");
```

```
                return SUCCESS;
            }
            else
            {
                setTip("转换失败");
                return ERROR;
            }
        }
    }
```

在上面程序中，可以看出 User 类型的属性需要进行类型转换。当 Struts 框架接受 HTTP 请求参数后，需要将这些参数封装成 User 对象。但是 Struts 2 提供的 OGNL 表达式允许开发者无须任何特殊处理，只需在定义表单域时使用 OGNL 即可。

2）定义表单页面 input.jsp，其代码如下。

```
<%@ page language="java" contentType="text/html; charset=GBK"%>
<html>
<head>
<title>利用 OGNL 进行类型转换</title>
</head>
<body>
<form action="login.action" method="post">
<table align="center" width="360">
   <caption><h3>利用 OGNL 进行类型转换</h3></caption>
   <tr>
        <!-- 该表单域封装的请求参数名为 user.name -->
        <td>用户名：<input type="text" name="user.name"/></td>
   </tr>
   <tr>
        <!-- 该表单域封装的请求参数名为 user.pass -->
        <td>密码：<input type="text" name="user.pass"/></td>
   </tr>
   <tr align="center">
        <td><input type="submit" value="转换"/>
        <input type="reset" value="重填" /></td>
   </tr>
</table>
</form>
</body>
</html>
```

上述页面代码请求的参数不是普通参数名，而是 OGNL 表达式的形式 user.name 和 user.pass，这时候就需要用 Java 类来处理，新建一个 user 类，其代码如下。

```
package lee;
public class User
```

```
{
    private String name;
    private String pass;
    public void setName(String name)
    {
        this.name = name;
    }
    public String getName()
    {
        return this.name;
    }
    public void setPass(String pass)
    {
        this.pass = pass;
    }
    public String getPass()
    {
        return this.pass;
    }
}
```

将上面的代码进行部署，运行后会得到如图 17-7 所示的效果。

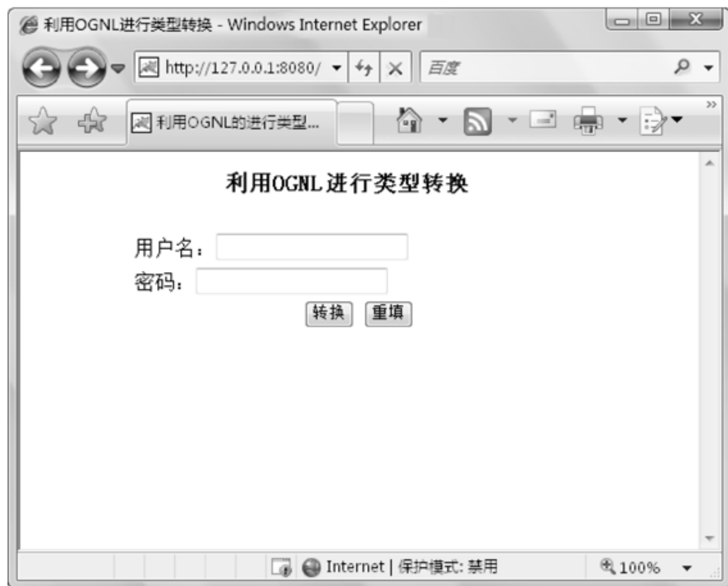


图 17-7　执行结果

多学一招

对于大多数数据来说，开发者不需要进行数据库类型转换，因为 Struts 2 可以完成大多的数据类型转换，在 Struts 2 中已经内置了类型转换机制。

17.3.2 自定义类型转换器

在开发程序时，有时需要将一个字符串转换成复合对象，这时需要用 Struts 2 自定义类型转换器。如果输入字符串“ab.rtt”，则需要一个 User 类型实例，其中 ab 作为实例的属性值，而 rtt 是 User 实例 Pass 的属性值。下面通过几段代码来说明自定义类型转换器的使用流程，代码保存在（光盘：源代码/第 17 章/17-3/）中，具体实现流程如下。

1）新建 LoginAction 类，其代码如下。

```
package lee;
import com.opensymphony.xwork2.Action;
public class LoginAction implements Action
{
   private User user;
   private String tip;
   public void setUser(User user)
   {
        this.user = user;
   }
   public User getUser()
   {
        return this.user;
   }
   public void setTip(String tip)
   {
        this.tip = tip;
   }
   public String getTip()
   {
        return this.tip;
   }
   public String execute() throws Exception
   {
        if (getUser().getName().equals("scott")
             && getUser().getPass().equals("tiger") )
        {
             setTip("转换成功");
             return SUCCESS;
        }
        else
        {
             setTip("转换失败");
             return ERROR;
        }
   }
}
```

2）新建 UserConverter 类，此类主要实现了类型转换器的功能，其代码如下。

```
package lee;
import java.util.Map;
import ognl.DefaultTypeConverter;
public class UserConverter extends DefaultTypeConverter
{
  //类型转换器必须重写 convertValue 方法，该方法需要完成双向转换
  public Object convertValue(Map context, Object value, Class toType)
  {
       //当需要将字符串向 User 类型转换时
       if (toType == User.class )
       {
            //系统的请求参数是一个字符串数组
            String[] params = (String[])value;
            //创建一个 User 实例
            User user = new User();
            //只处理请求参数数组第一个数组元素，并将该字符串以英文逗号分割成两个字符串
            String[] userValues = params[0].split(",");
            //为 User 实例赋值
            user.setName(userValues[0]);
            user.setPass(userValues[1]);
            //返回转换来的 User 实例
            return user;
       }
       else if (toType == String.class )
       {
            //将需要转换的值强制类型转换为 User 实例
            User user = (User) value;
           return "<" + user.getName() + "," + user.getPass() + ">";
       }
       return null ;
  }
}
```

3）新建 user 类，此类的实现文件代码十分简单，功能是处理 name 和 pass 的取值和传递值，其代码如下。

```
package lee;
public class User
{
  private String name;
  private String pass;
  public void setName(String name)
  {
       this.name = name;
  }
```

```
        public String getName()
        {
            return this.name;
        }
        public void setPass(String pass)
        {
            this.pass = pass;
        }
        public String getPass()
        {
            return this.pass;
        }
    }
```

4）新建文件 input.jsp，其页面代码如下。

```
<%@ page language="java" contentType="text/html; charset=GBK"%>
<html>
<head>
<title>局部类型转换器</title>
</head>
<body>
<form action="login.action" method="post">
    <table align="center" width="360">
    <caption><h3>局部类型转换器</h3></caption>
        <tr align="center">
            <td>用户信息的用户名和密码以英文逗号隔开</td>
        </tr>
        <tr>
            <td>请输入用户信息：<input type="text" name="user"/></td>
        </tr>
        <tr align="center">
            <td><input  type="submit"  value="转换"/><input  type="reset" 
value="重填" /></td>
        </tr>
    </table>
</form>
</body>
</html>
```

5）新建文件 welcome.jsp，其页面代码如下。

```
<%@ page language="java" contentType="text/html; charset=GBK"%>
<%@taglib prefix="s" uri="/struts-tags"%>
<html>
    <head>
        <title>转换成功</title>
```

```
        </head>
        <body>
            转换成功!<br>
              用户名为:<s:property value="user.name"/><br>
              密码为：<s:property value="user.pass"/><br>
        </body>
    </html>
```

在浏览器中输入调试地址，运行后的效果如图 17-8 所示。

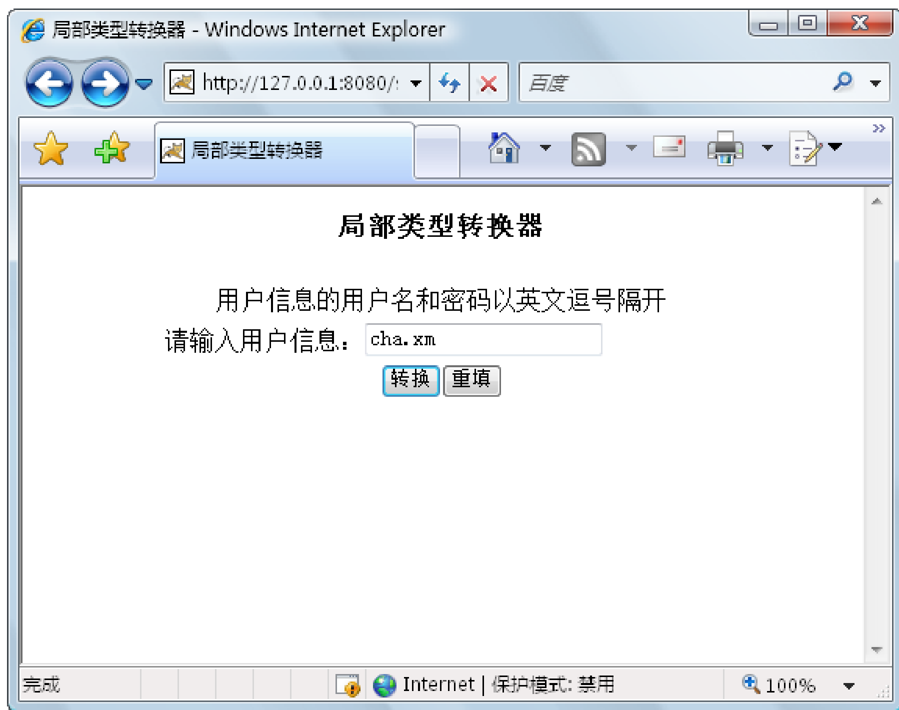


图 17-8 类型转换

17.4 文件的上传和下载

文件的上传和下载是 Struts 2 的重要内容，也是 Java Web 中比较常见的操作。在本节的内容中，将详细讲解文件的上传和下载功能的实现。

17.4.1 文件的上传

在博客系统、邮箱系统或者企业网站中，都需要将本地的文件上传到远程服务器。文件上传功能十分重要，究竟应该如何实现呢？下面通过一个实例进行讲解。

实例 36：实现文件上传

本实例实现了文件上传功能，其代码保存在（光盘：源代码/第 17 章/17-4/）中，具体实现流程如下。

1）新建一个类 UploadAction，其代码如下。

```
package lee;

import com.opensymphony.xwork2.Action;
import org.apache.struts2.ServletActionContext;
import java.io.File;
import java.io.*;
import com.opensymphony.xwork2.ActionSupport;
public class UploadAction extends ActionSupport
{
   //封装文件标题请求参数的属性
   private String title;
   //封装上传文件域的属性
   private File upload;
   //封装上传文件类型的属性
   private String uploadContentType;
   //封装上传文件名的属性
   private String uploadFileName;
   //直接在 struts.xml 文件中配置的属性
    private String savePath;
   //接受 struts.xml 文件配置值的方法
    public void setSavePath(String value)
   {
       this.savePath = value;
    }
   //返回上传文件的保存位置
    private String getSavePath() throws Exception
   {
       return ServletActionContext.getRequest().getRealPath(savePath);
    }
   //文件标题的 setter 和 getter 方法
   public void setTitle(String title)
   {
        this.title = title;
   }
   public String getTitle()
   {
        return (this.title);
   }
   //上传文件对应文件内容的 setter 和 getter 方法
   public void setUpload(File upload)
   {
        this.upload = upload;
   }
   public File getUpload()
   {
```

```
            return (this.upload);
        }
        //上传文件的文件类型的 setter 和 getter 方法
        public void setUploadContentType(String uploadContentType)
        {
            this.uploadContentType = uploadContentType;
        }
        public String getUploadContentType()
        {
            return (this.uploadContentType);
        }
        //上传文件的文件名的 setter 和 getter 方法
        public void setUploadFileName(String uploadFileName)
        {
            this.uploadFileName = uploadFileName;
        }
        public String getUploadFileName()
        {
            return (this.uploadFileName);
        }
         public String execute() throws Exception
        {
            //以服务器的文件保存地址和原文件名建立上传文件输出流
            FileOutputStream fos = new FileOutputStream(getSavePath()
                + "\\" + getUploadFileName());
            FileInputStream fis = new FileInputStream(getUpload());
            byte[] buffer = new byte[1024];
            int len = 0;
            while ((len = fis.read(buffer)) > 0)
            {
                fos.write(buffer , 0 , len);
            }
           return SUCCESS;
         }
    }
```

2）新建文件 upload.html，具体代码如下。

```
    <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=GBK" />
    <title>简单的文件上传</title>
    </head>
    <body>
    <form action="upload.action" method="post" enctype="multipart/form-data">
       文件标题：<input type="text" name="title" /><br>
       选择文件：<input type="file" name="upload" /><br>
```

```
        <input value="上传" type="submit" />
    </form>
    </body>
    </html>
```

3）新建文件 succe.jsp，具体代码如下。

```
    <%@ page language="java" contentType="text/html; charset=GBK"%>
    <%@taglib prefix="s" uri="/struts-tags"%>
    <html>
       <head>
          <title>上传成功</title>
       </head>
       <body>
          上传成功!<br>
           文件标题:<s:property value=" + title"/><br>
           文件为：<img src="<s:property value="'upload/'+uploadFileName"/>"/>
<br>
       </body>
    </html>
```

4）设置配置文件 Struts.xml，具体代码如下。

```
    <?xml version="1.0" encoding="gbk"?>
    <!DOCTYPE struts PUBLIC
      "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
      "http://struts.apache.org/dtds/struts-2.0.dtd">
    <struts>
      <constant name="struts.custom.i18n.resources" value="globalMessages"/>
      <constant name="struts.i18n.encoding" value="GBK"/>
      <package name="lee" extends="struts-default">
               <action name="upload" class="lee.UploadAction">
                              <param name="savePath">/upload</param>
                              <result>/succ.jsp</result>
           </action>
           <action name="">
               <result>.</result>
           </action>
      </package>
    </struts>
```

5）修改文件 web.xml，修改后的代码如下。

```
    <?xml version="1.0" encoding="GBK"?>
    <web-app xmlns="http://java.sun.com/xml/ns/javaee"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
      http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
```

```
            <filter>
            <filter-name>struts2</filter-name>
            <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</
filter-class>
        </filter>
            <filter-mapping>
            <filter-name>struts2</filter-name>
            <url-pattern>/*</url-pattern>
        </filter-mapping>

        <filter>
           <filter-name>struts-cleanup</filter-name>
           <filter-class>org.apache.struts2.dispatcher.ActionContextCleanUp</
filter-class>
         </filter>
        <filter-mapping>
           <filter-name>struts-cleanup</filter-name>
           <url-pattern>/*</url-pattern>
         </filter-mapping>
      </web-app>
```

6）在“webroot”目录下新建一个“upload”文件夹，然后启动服务器，输入地址浏览 upload.html 页面。运行后选择一个图片，然后输入文件标题，单击“上传”按钮后会将此图片文件上传到服务器。执行效果如图 17-9 所示。

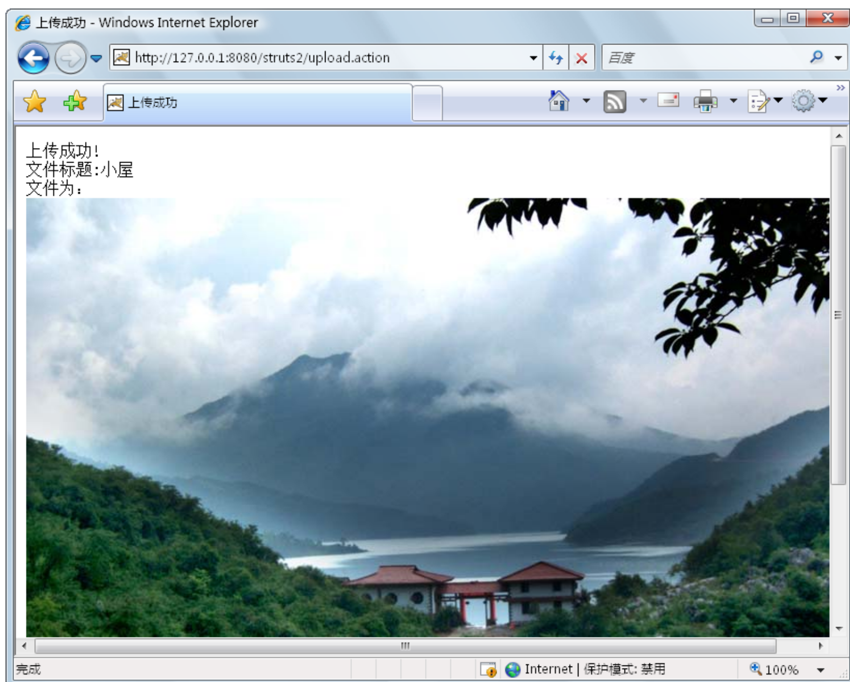


图 17-9　上传文件

 多学一招

虽然上传文件的过程十分简单，但是初学者一定要注意对权限和文件上传大小的控制，因为这跟服务器安全息息相关。

17.4.2 文件的下载

接下来讲的下载不是指静态下载，不是单击超级链接就能实现下载功能的那种。在此讲的下载需要有一定的权限，在很多电影网站中，用户需要一定权限才能下载。在 CSDN 论坛里，有些资源也需要一定权限才能下载。在 Java Web 中可以通过 Struts 2 完成上述需要权限的下载功能，下面通过几段代码（光盘：源代码/第 17 章/17-4/）讲解此类下载功能的实现过程。

1）新建一个 AuthorityDownAction 类，具体代码如下。

```
package lee;
import java.io.InputStream;
import org.apache.struts2.ServletActionContext;
import com.opensymphony.xwork2.Action;
import com.opensymphony.xwork2.ActionContext;
import java.util.Map;
public class AuthorityDownAction implements Action
{
  private String inputPath;
  public void setInputPath(String value)
  {
        inputPath = value;
  }
  public InputStream getTargetFile() throws Exception
  {
        //ServletContext 提供 getResourceAsStream()方法
        //返回指定文件对应的输入流
        return ServletActionContext.getServletContext()
              .getResourceAsStream(inputPath);
  }
  public String execute() throws Exception
  {
        //取得 ActionContext 实例
        ActionContext ctx = ActionContext.getContext();
        //通过 ActionContext 访问用户的 HttpSession
        Map session = ctx.getSession();
        String user = (String)session.get("user");
        //判断 Session 里的 user 是否通过检查
        if ( user !=  null && user.equals("admin"))
        {
              return SUCCESS;
        }
        ctx.put("tip" , "您还没有登录，或者登录的用户名不正确，请重新登录！");
```

```
            return LOGIN;
        }
    }
```

2）新建类 FileDownloadAction，具体代码如下。

```
    package lee;
    import java.io.InputStream;
    import org.apache.struts2.ServletActionContext;
    import com.opensymphony.xwork2.*;
    public class FileDownloadAction extends ActionSupport
    {
        //该属性是依赖注入的属性，可以在配置文件中动态指定该属性值
        private String inputPath;
        //依赖注入该属性值的 setter 方法
        public void setInputPath(String value)
        {
            inputPath = value;
        }
        /*
        定义一个返回 InputStream 的方法，
        该方法将作为被下载文件的入口，
        且需要配置 stream 类型结果时指定 inputName 参数，
        inputName 参数的值就是方法去掉 get 前缀、首字母小写的字符串
        */
        public InputStream getTargetFile() throws Exception
        {
            //ServletContext 提供 getResourceAsStream()方法
            //返回指定文件对应的输入流
            return ServletActionContext.getServletContext()
                .getResourceAsStream(inputPath);
        }
    }
```

3）新建类 LoginAction，具体代码如下。

```
    package lee;
    import java.io.InputStream;
    import org.apache.struts2.ServletActionContext;
    import com.opensymphony.xwork2.Action;
    import com.opensymphony.xwork2.ActionContext;
    public class LoginAction implements Action
    {
        private String user;
        private String pass;
        public void setUser(String user)
        {
            this.user = user;
        }
```

```
        public String getUser()
        {
            return (this.user);
        }
            public void setPass(String pass)
        {
            this.pass = pass;
        }
        public String getPass()
        {
            return (this.pass);
        }
        public String execute(){
            ActionContext.getContext().getSession()
                .put("user" , getUser());
            return SUCCESS;
        }
    }
```

4）设置配置文件 Struts.xml，其页面代码如下。

```
    <?xml version="1.0" encoding="GBK"?>
    <!DOCTYPE struts PUBLIC
      "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
      "http://struts.apache.org/dtds/struts-2.0.dtd">
    <struts>
      <!-- 配置 Struts2 国际化资源文件的 baseName -->
      <constant name="struts.custom.i18n.resources" value="globalMessages"/>
      <!-- 配置 Struts2 应用的编码集 -->
      <constant name="struts.i18n.encoding" value="GBK"/>
      <package name="lee" extends="struts-default">
          <action name="download" class="lee.FileDownloadAction">
              <!-- 指定被下载资源的位置 -->
              <param name="inputPath">\images\xiaowu.jpg</param>
              <!-- 配置结果类型为 stream 的结果 -->
              <result name="success" type="stream">
                  <!-- 指定下载文件的文件类型 -->
                  <param name="contentType">image/jpg</param>
                  <!--指定由 getTargetFile()方法返回被下载文件的 InputStream-->
                  <param name="inputName">targetFile</param>
                  <param  name="contentDisposition">filename="xiaowu.
jpg"</param>
                  <!-- 指定下载文件的缓冲大小 -->
                  <param name="bufferSize">4096</param>
              </result>
          </action>
          <action name="download2" class="lee.AuthorityDownAction">
              <!-- 定义被下载文件的物理资源 -->
              <param name="inputPath">\images\xiaowu.zip</param>
```

```
                <result name="success" type="stream">
                     <!-- 指定下载文件的文件类型 -->
                     <param name="contentType">application/zip</param>
                     <!--指定由getTargetFile()方法返回被下载文件的InputStream -->
                     <param name="inputName">targetFile</param>
                     <param  name="contentDisposition">filename= "xiaowu.
zip" </param>
                     <!-- 指定下载文件的缓冲大小 -->
                     <param name="bufferSize">4096</param>
                </result>
                <!-- 定义一个名为 login 的结果 -->
                 <result name="login">/input.jsp</result>
           </action>
           <action name="login" class="lee.LoginAction">
                <result>/stuts2Down.html</result>
           </action>
           <action name="">
                <result>.</result>
           </action>
      </package>
   </struts>
```

接下来需要实现两个静态页面和一个 JSP 登录页面，因为和前面的代码一样，所以在此将不展示具体代码。执行之后，如果输入的用户名和密码不正确，就不能下载压缩文件。单击下载压缩文件超级链接后，会得到如图 17-10 所示的页面。如果用户输入的用户名和密码正确（用户名为 admin，密码为空），就会得到如图 17-11 所示效果。

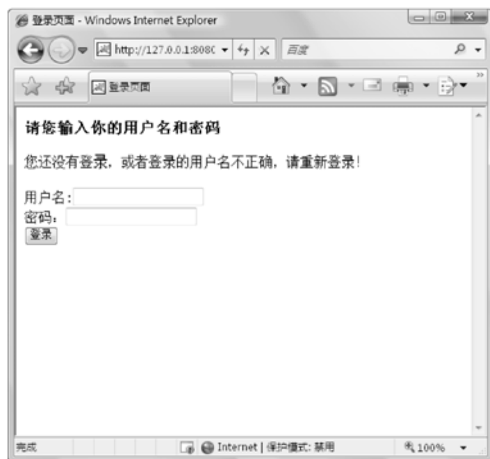


图 17-10　无权限下载

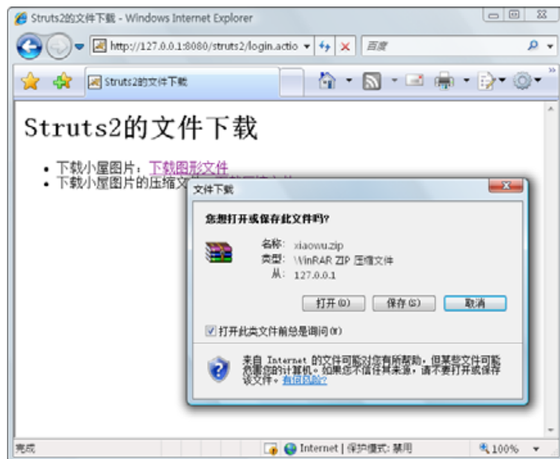


图 17-11　可以下载

17.5　疑难问题解析

在本章的内容中，详细讲解了 Struts 2 架构的基本知识。本节中，将对本章中比较难理

解的问题进行讲解。

读者疑问：在前面讲解了拦截器，常听一些有经验的程序员讲，要求拦截器在无状态下执行，这是为什么呢？

解答：要求拦截器是无状态的原因是 Struts 2 不能保证为每一个请求或者 action 创建一个实例，如果拦截器带有状态，就会并发许多未知的问题。在前面的章节已经讲解过，所有的 Struts 2 的拦截器都直接或间接实现接口 com.opensymphony.xwork2.interceptor.Interceptor。除此之外，大家可能更喜欢继承类 com.opensymphony.xwork2.interceptor.AbstractInterceptor。

读者疑问：文件的上传是 web 工程经常用到的一个工具，实现很简单，Struts 2 是如何实现的呢，它的原理在哪里？它经历一个怎样的过程呢？

解答：Struts 2 是通过 Commons FileUpload 文件上传。Commons FileUpload 通过将 HTTP 的数据保存到临时文件夹，然后 Struts 使用 FileUpload 拦截器将文件绑定到 Action 的实例中，我们就能够以本地文件方式的操作浏览器上传的文件。



职场点拨——教会徒弟饿死师傅

今天总监给 HR 打了招呼，让他在招聘事务上带带我。至于细节安排，比如要在多长时间内，达到怎样的阶段培训目标，并没有详细交代。这可就苦了我了，作为人力资源领域的一个新人，我完全没有方向，连提问都不知道怎么提。HR 本来就不愿意教我，他让手下的专员小小鸟给我简单介绍了一下区域 HR 负责招聘的岗位的薪酬结构，还有就是公司的基本福利制度，不过个把小时，算完成了薪酬部分对我的培训。

小小鸟讲完，客气地问我有什么需要他进一步解答的地方，我刚接触 HR，问题也无从提起，勉强问了几个问题，小小鸟事先得到 HR 的授意，都回答说是属于保密的内容，没法告诉我。末了，我只得说暂时没有问题。

自从我学习招聘后，越是着急学习 HR 的知识，HR 就越是对我防卫，特别是碰到想学的薪酬福利相关内容的时候，HR 抑制不住的反感。

这是小菜的经历，相信有很多职场人员也会有类似的经历，为什么从前辈那里学点东西这么困难呢？这和 HR 的经历有关。

在 HR 加入公司之前，大部分时间在中小公司工作，成长得比较艰辛，过去他向同行学每一点东西都不容易，别说主动教他，不想法阻挠他学就算不错了。现在这个小菜倒好，今天问这个，明天问那个，好像教他 HR 是别的 HR 经理应尽的工作职责一样，全不懂规矩，教会徒弟，饿死师傅不成吗？

所以当小菜提出来想了解公司的新 HR 系统软件后，HR 马上私下里交待手下的专员小小鸟说："不需要把系统的权限放给区域 HR，否则会有太多人可以进入系统，对系统的维护不好。小菜以后遇到任何从系统里调资料或者往系统里输入资料的需求，必须填写单子交给你，经我批准后，一概由你这边办理。"

在现实职场中，此类事情太多了。因为同事之间存在竞争关系，即使是领导让老资格的同事教你东西，虽然他们表面没有任何问题，但是心底里是极不情愿的。因为你一旦学会了，他们的地位就会下降。希望读者仔细品味这个真实的故事，能从中感悟一些东西。

第 18 章　Spring 详解

虽然 Struts 2 十分优秀，但是也存在一些不足。后来随着 Spring 的出现，弥补了 Struts 2 框架的很多不足。在本章的内容中，将详细讲解 Sping 框架的基本知识。通过本章能学到如下知识。

- Spring 的依赖注入。
- Spring 的容器。
- 职场点拨——同事间的互补。

2011 年 XX 月 XX 日，阴

公司最近要组织一个开发团队，项目经理让我挑选团队成员。我重点考虑了 A、B 和 C。

A：比较有背景，不但编程经验丰富，还做过知名企业的项目经理，有管理经验。

B：工作经验丰富，从事编程 10 年了，一直是软件工程师，看着与世无争的样子。

C：有一点非主流，技术一般，但是沟通交流能力一流。

我思考了半天，决定选 A 和 B。并总结道：A 和 B 技术过硬，必能大大缩短开发工期。

但是最后结果出来后老大选择了 A 和 C，小菜很困惑。老大说："我们这个项目，客户很难缠，C 的沟通能力较好，对整个项目的进展能有很大帮助！"



一问一答

小菜："今天老大推翻了我的选择，他认为 A 和 C 能形成互补！"

Wisdom："嗯，同事间互补很重要，本章的最后将详细介绍具体原因！"

小菜："嗯，言归正传，本章的 Spring 很重要吗？"

Wisdom："当然重要，它和 Struts 2 也是互补的！就像 A 和 C 是一对好的拍档，A 技术过硬，C 沟通力强，两者实现了互补。而本章所讲的 Spring 实现了和 Struts 2 的互补。Spring 是 Java EE 的重要框架，通过它们，用户省去很多麻烦。"

18.1　认识 Spring

Spring 是 Java EE 的重要框架，通过使用 Spring，开发人员可以省去很多麻烦。在使用 Spring 之前需要先下载并安装它，图 18−1 是下载 Spring 的主页。



图 18-1　Spring 框架

18.2　Spring 基础

Spring 框架可以深入任何框架，并且在业务层表现特别优秀，所以在现实中经常将它用于业务层。Spring 是一个开源框架，由 Rod Johnson 创建，它是为了解决企业应用开发的复杂性而创建的。Spring 使用基本的 JavaBean 来完成以前只可能由 EJB 完成的事情。Spring 的用途不仅限于服务器端的开发，任何 Java 应用都可以从 Spring 中受益。Spring 能够解决企业应用开发的复杂性，可以使用基本的 JavaBean 代替 EJB，并提供了更多的企业应用功能。

18.2.1　为什么需要 Spring

Spring 的突出优点如下。

- Spring 定位的领域是众多流行的 Framework 没有关注的，它能提供一种方法管理我们的业务对象。
- Spring 是全面的和模块化的。Spring 有严格的分层体系结构，这样可以选择使用它的任何部分，它的架构仍然是内在稳定的。
- Spring 从底部开始帮助我们编写易于测试的代码，是用于测试驱动工程的理想 Framework。

- ❑ Spring 不需要一个以上的 Framework，它是潜在的一站式解决方案，定位于与典型应用相关的大部分基础结构。Spring 涉及其他 Framework 中没有考虑的内容。

18.2.2　Spring 将改变 Java EE

Spring 会给一个 Java 工程带来很多好处。总结起来，Spring 在 Java EE 中有如下优点。

- ❑ 低侵入式设计，代码污染极低。
- ❑ 独立于各种应用服务器，可以真正实现 Write Once,Run Anywhere 的承诺。
- ❑ Spring 的 DI 机制降低了业务对象替换的复杂性。
- ❑ 开发者可自由选用 Spring 框架的部分或全部。

18.2.3　如何获得 Spring

Spring 是一个免费的开源框架，读者可以根据自己需要去选择并使用它。获得 Spring 框架的方法很简单，其具体操作流程如下。

1）打开浏览器，在浏览器中输入网址 http://www.springframework.org/ download，建议下载 Spring 的最新稳定版本，例如，Spring Framework 2.5.6，在这个压缩包中不仅包含了 Spring 的开发包，而且包含 Spring 编译和运行所依赖的第三方类库。如图 18-2 所示。



图 18-2　spring 首页

2）单击导航栏中的“download”中的超级链接，然后选择需要下载的版本，输入相关的信息，单击“Submit”按钮后即可下载 Spring，如图 18-3 所示。

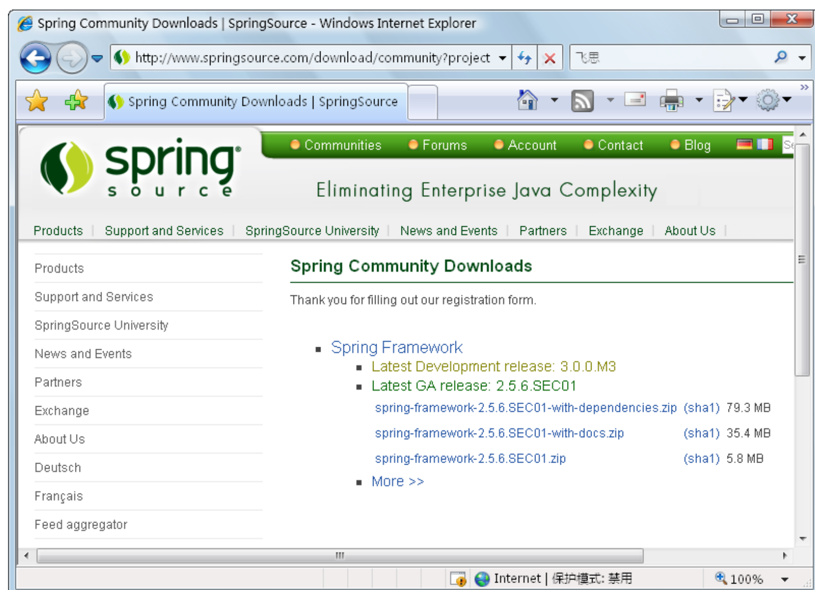


图 18-3　下载 Spring

18.2.4　部署 Spring

Spring 下载完成后，为了开发方便，用户可以将其部署到开发的 IDE 中去。执行 Spring 的方法很简单，只需要将需要的 Jar 文件复制到 Web 站点下的“WEB-INF/lib”路径下即可，它与 Struts 2 和 Hibernate 的配置十分相似，复制进去后需要进行简单的设置。部署 Spring 的具体操作步骤如下。

1）解压下载的文件，完成后得到如图 18-4 所示的文件。

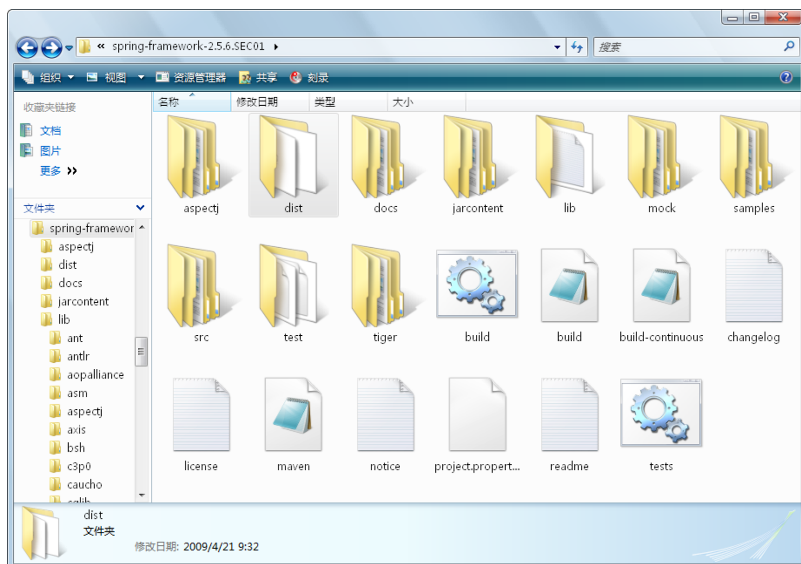


图 18-4　Spring 的结构图

2）启动 MyEclipse，将需要的 Jar 文件复制到项目中的“WEB-INF”目录下。在本书光

盘中，也提供了这些 Jar 文件（光盘：源代码/第 18 章/spring/）。用户不必自己去查找，可以直接复制使用，此时在项目中会多了许多配置文件，如图 18-5 所示。

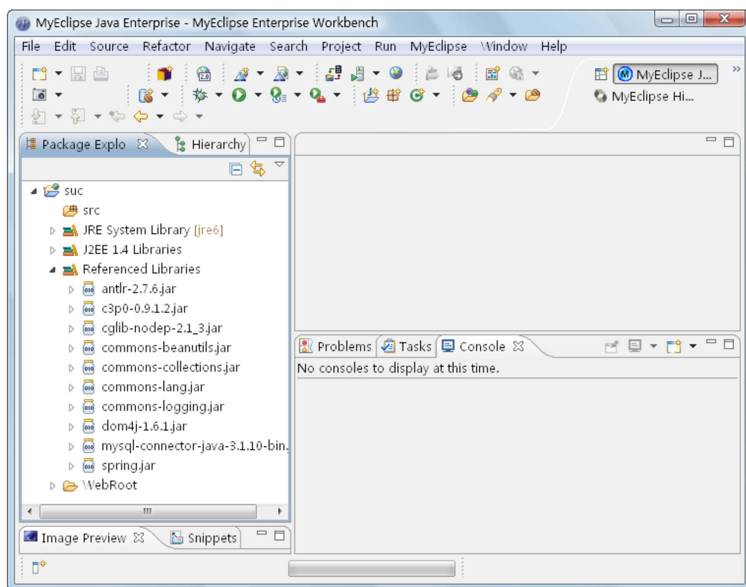


图 18-5　需要的 jar 文件

3）复制后进行刷新，选择新建的项目，然后单击鼠标右键，在弹出的快捷菜单中选择“MyEclipse/Add Spring Capabilities”命令，如图 18-6 所示。

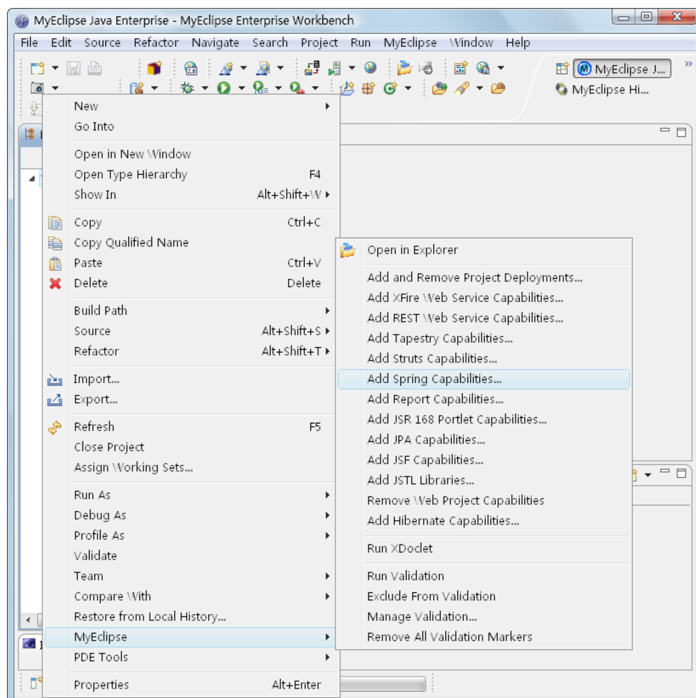


图 18-6　选择命令

4）打开“Add Spring Capabilities”窗口，可以根据自己的需要设置 jar 文件。设置完成后单击“Next”按钮，如图 18-7 所示。

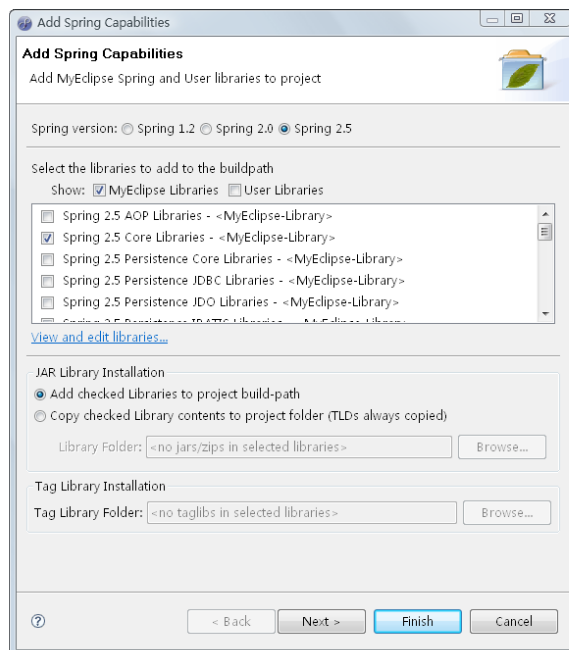


图 18-7　设置 Spring 2.5 文件

5）单击“Next”按钮，指定在什么位置建立一个 Spring 文件，这里保持默认不变，单击“Finsh”按钮，如图 18-8 所示。

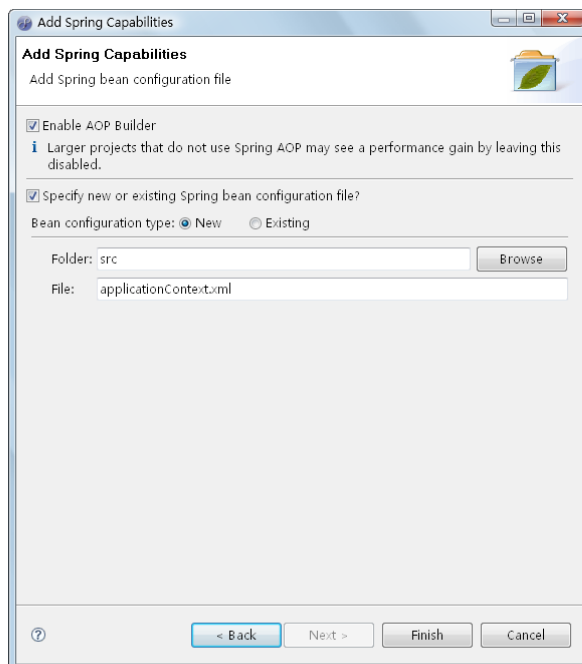


图 18-8　建立 Spring 文件

6）建立完成后，可以在 MyEclipse 中见到新建的 Spring 文件，如图 18-9 所示。

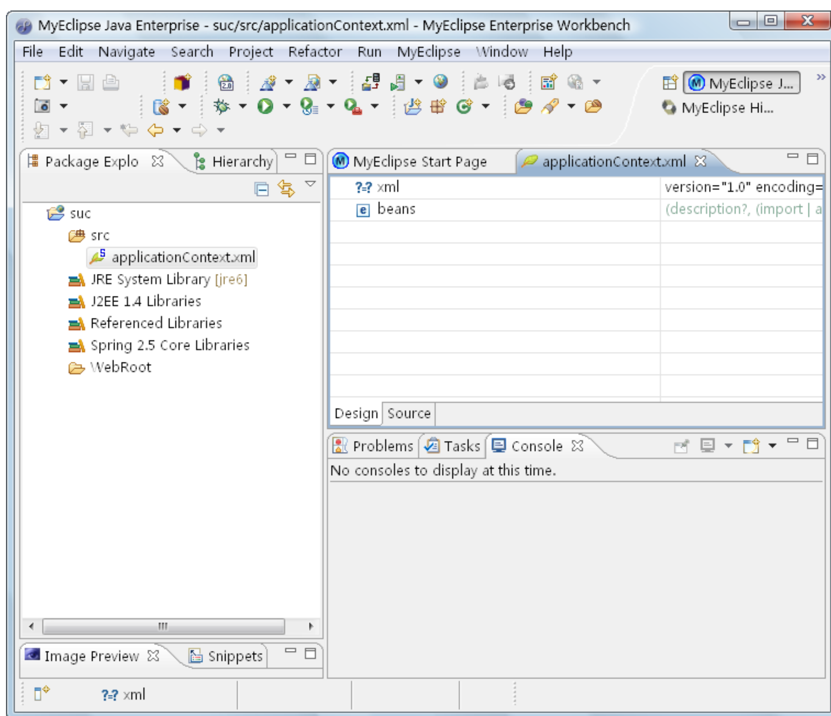


图 18-9　新建的 Spring 文件

18.2.5　第一个 Spring 文件

接下来将讲解一个 Spring 程序，让读者理解 Spring 的执行流程，以及它的特点。代码保存在（光盘：源代码/第 18 章/18-1/）中，下面开始讲解实现流程。

1）新建一个名为“bean.xml”的文件，其页面代码如下。

```
<?xml version="1.0" encoding="GBK"?>
<!-- Spring 配置文件的 DTD 信息 -->
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN 2.0//EN"
   "http://www.springframework.org/dtd/spring-beans-2.0.dtd">
<!--配置文件的根元素 -->
<beans>
   <!-- 定义 Bean，id 是 chineseone, class 指定实现类 -->
   <bean id="chinese" class="lee.Chinese">
         <!--  property 元素用来指定需要容器注入的属性，axe 属性需要容器注入
                此处是设值注入，因此 Chinese 类必须拥有 setAxe 方法 -->
         <property name="axe" ref="steelAxe"/>
   </bean>
   <!-- 定义 steelAxe bean -->
   <bean id="steelAxe" class="lee.SteelAxe"/>
</beans>
```

2）新建一个 BeanTest 类，其代码如下。

```
package leeone;//包
import java.io.*;//插入包
import org.springframework.core.io.ClassPathResource; //插入包
import org.springframework.beans.factory.xml.XmlBeanFactory;//插入包
public class BeanTestone//声明一个类
{
   public static void main(String[] args)throws Exception
   {
         //以 CLASSPATH 路径下的配置文件，创建 ClassPathResource 实例
         ClassPathResource isr = new ClassPathResource("bean.xml");
         //以 Resource 对象作为参数，创建 BeanFactory 的实例
         XmlBeanFactory factory = new XmlBeanFactory(isr);
         System.out.println("程序已经实例化 BeanFactory...");
         Person p = (Person)factory.getBean("chinese");
         System.out.println("程序中已经完成了 chinese Bean 的实例化...");
         p.useAxe();
   }
}
```

3）新建一个 SteelAxe 类，其代码如下。

```
package lee;
public class SteelAxeone implements Axe
{
   //默认构造器
   public SteelAxeone()
   {
       System.out.println("实例化一般依赖 Bean：SteelAxeone 实例...");
   }
   //实现 Axe 接口的 chop 方法
   public String chop()
   {
         return "很快";
   }
}
```

4）新建一个 Chinese 类，其代码如下。

```
package lee;
public class Chineseone implements Personone
{
   private Axe axe;
   //构造器
   public Chinese()
   {
```

```
            System.out.println("Spring 实例化主调 Bean：Chinese 实例...");
        }
        //注入 setter 方法
        public void setAxeone(Axe axe)
        {
            System.out.println("Spring 执行依赖关系注入...");
            this.axe = axe;
        }
        //实现 useAxe 方法
        public void useAxeone()
        {
            System.out.println(axe.chop());
        }
    }
```

5）新建接口 interface Person，其代码如下。

```
    package lee;
    public interface Person
    {
        //Person 接口里定义一个使用斧子的方法
        public void useAxe();
    }
```

6）新建文件 bulid.xml，其代码如下。

```
    <?xml version="1.0" encoding="GBK"?>
    <project name="hibernte" basedir="." default="">
        <property name="src" value="src"/>
        <property name="dest" value="classes"/>
         <path id="classpath">
            <fileset dir="..\..\lib">
                <include name="*.jar"/>
            </fileset>
            <pathelement path="${dest}"/>
         </path>
        <target name="compile" description="Compile all source code">
             <delete dir="${dest}"/>
             <mkdir dir="${dest}"/>
             <copy todir="${dest}">
              <fileset dir="${src}">
                       <exclude name="**/*.java"/>
                  </fileset>
             </copy>
            <javac destdir="${dest}" debug="true"
                deprecation="false" optimize="false" failonerror="true">
                <src path="${src}"/>
```

```
                <classpath refid="classpath"/>
            </javac>
        </target>
        <target  name="run"  description="Run  the  main  class"  depends=
"compile">
            <java classname="lee.BeanTest" fork="yes" failonerror="true">
                <classpath refid="classpath"/>
            </java>
        </target>
    </project>
```

上面的代码十分简单，执行后可以在调试栏看到下面的信息。

```
程序已经实例化 BeanFactory
Spring 是主调 Bean：Chinese 实例……
Spring 是实例化依赖 Bean：SteelAxe 实例
Spring 执行依赖关系注入……
程序中已经完成了 Chinese Bean 的实例化……
```

在上面的调试栏信息中，可以看出 Spring 所经历的过程。

18.3 Spring 的依赖注入

在 Java 项目开发中，Spring 虽不是依赖注入的创始者，但是它对开发 Java 项目有深远的意义，下面将学习依赖注入的基本知识。

18.3.1 什么是依赖注入

依赖注入是指程序在运行的过程中，需要调用一个对象协助，不需要创建调用者，而直接依赖外部的支持与帮助。在 Spring 框架中，对调用和被调用者几乎没有任何要求，完全支持 POJO（Plain Old Java Objects，简单的 Java 对象）之间的依赖关系。依赖注入有两种，介绍如下。

- 设值注入：在 IOC（Inversion of Control，控制反转）容器使用属性的 Setter 方法来注入被依赖的实例。
- 构造注入：IOC 容器使用构造器来注入被依赖的实例。

依赖注入是Spring框架的核心。通过使用依赖注入，当对象被创建的时候，由一个调控系统内所有对象的外界实体，将其所依赖的对象的引用传递给它。简单一点说，就是依赖被注入到对象中。对于初学者来说，很难理解依赖注入。在理解依赖注入之前，在此给读者打一个简单的比方，如果遇到这样一个问题应该如何解决：一个人需要一把斧子，人是 Java 实例中的调用者，斧子是被调用者，一个人需要一把斧子可能有如下 3 种情况。

- 在原始社会里，一切都很简单，没有社会分工。需要斧子的人(调用者)只能靠自己去打磨一把斧子（被调用者）。对应 Java 的设计情形就是 Java 程序里的调用者自己创建被调用者。

- 随着科技的发展，进入工业社会，出现了工厂。斧子可能不由普通人完成，而是被工厂生产出来，此时需要斧子的人（调用者）就要找到工厂，购买斧子，无须关心斧子的制造过程。它对应起来，就和 Java 程序的简单工厂的设计模式类似。
- 当人类进入高度文明的时代，按需分配社会：需要斧子的人不需要找到工厂，坐在家里发出需要斧子的需求。斧子就自然出现在他面前。这对应 Spring 的依赖注入。

依赖注入有如下 3 种情况。

1）Java 实例的调用者创建被调用的 Java 实例，肯定会要求被调用的 Java 类出现在调用者的代码中。如果无法实现二者之间的高度配合，则效率会极低，现实中很少应用这种方式。

2）调用者不必关心被调用者的具体实现过程，只要找到符合某种标准（接口）的实例即可。此时调用的代码需要面向接口编程，可以让调用者和被调用者解耦，这也是工厂模式大量流行的原因。但这需要调用者自己定位工厂，调用者与特定工厂耦合在一起。

3）调用者不必定位工厂，程序运行到需要被调用时，系统自动提供被调用者实例。调用者和被调用者都处于 Spring 的管理下，二者之间的依赖关系由 Spring 提供。

18.3.2　设置注入

设置注入的方式比较简单和直观，在 Spring 的依赖注入中会大量使用设置注入。设置注入是通过 Setter 方法传入被调用者的实例，下面将通过几段代码来讲解设置注入的实现流程。

1）定义 Person 接口和 Axe 接口，Person 接口的实现代码（光盘：源代码/第 18 章/18-2/）如下。

```
//定义 Person 接口
public interface Personone
{
  //Person 接口里定义一个方法
  public void useAxe();
}
Axe 接口的实现代码【光盘：源代码/第 18 章/18-2/】如下：
//定义 Axe 接口
public interface Axeone
{
  //Axe 接口定义一个方法
  public void chop();
}
```

2）定义一个接口，实现 Person 接口里的 useAxe 方法，其代码（光盘：源代码/第 18 章/18-2/）如下。

```
//定义一个类 Chinese 用来实现 Person 接口
public class Chineseone implements Personone
{
  //面向 Axe 接口编程，而不是具体的实现类
```

```
        private Axe axe;
        //默认的构造器
        public Chinese()
        {
      }
        //设置所需的 setter 方法
        public void setAxe(Axe axe)
        {
      this.axe = axe;
        }
        //实现 useAxe 方法
        public void useAxe()
        {
          System.out.println(axe.chop());
        }
      }
```

3）用一个类来实现 Axe 接口，其代码（光盘：源代码/第 18 章/18-2/）如下。

```
      //Axe 的第一个实现类 StoneAxeone
      public class StoneAxeone implements Axe
      {
        //默认构造器
        public StoneAxe()
        {
      }
        //实现 Axe 接口的 chop 方法
        public String chop()
        {
      return "设置注入";
        }
      }
```

通过本书前面内容的学习，已经了解了 Spring 框架和 Struts 框架的基本知识，它们都有自己的配置文件。同样，上面的文件也需要配置，其配置代码（光盘：源代码/第 18 章/18-2/）如下。

```
      <!-- 下面是标准的 XML 文件头 -->
      <?xml version="1.0" encoding="gb2312"?>
      <!-- 下面一行定义 Spring 的 XML 配置文件的 dtd -->
      "http://www.springframework.org/dtd/spring-beans.dtd">
      <!-- 以上三行对所有的 Spring 配置文件都是相同的 -->
      <!-- Spring 配置文件的根元素 -->
      <BEANS>
        <!—定义第一 bean，该 bean 的 id 是 chineseone,class 指定该 bean 实例的实现类-->
        <BEAN class=lee.Chinese id=chineseone>
        <!-- property 元素用来指定需要容器注入的属性，axe 属性需要容器注入此处是设置注
```

```
入，因此 Chinese 类必须拥有 setAxe 方法 -->
        <property name="axe">
        <!-- 此处将另一个 bean 的引用注入给 chinese bean -->
        <REF local=""stoneAxe"/">
        </property>
        </BEAN>
        <!-- 定义 stoneAxe bean -->
        <BEAN class=lee.StoneAxe id=stoneAxe />
        </BEANS>
```

从上面的配置文件可以看出，使用 Spring 框架可以灵活地管理 Bean。Spring 与 Bean 之间的关系依赖存放在配置文件中，而不是依靠代码完成。通过配置文件的指定，Spring 能精确地为每一个 Bean 注入属性，因此配置文件的 Bean 的 class 元素必须是真正的实现类。Spring 会自动管理每一个 Bean 定义的 Property 元素定义，Spring 会执行无参数的构造器，创建默认的 Bean 实例。每个 Bean 的 id 属性是该 Bean 的唯一标识，程序通过 id 属性访问 Bean，Bean 与 Bean 的依赖关系也是通过 id 属性完成的。下面是上面类和接口的主程序，具体代码如下。

```
        public class BeanTestone
        {
          //主方法，程序的入口
          public static void main(String[] args)throws Exception
          {
            //因为是独立的应用程序，显式地实例化 Spring 的上下文。
              ApplicationContext  ctx  =  new  FileSystemXmlApplicationContext
("bean.xml");
            //通过 Person Bean 的 id 来获取 bean 实例，面向接口编程，因此
            //此处强制类型转换为接口类型
            Person p = (Person)ctx.getBean("chinese");
            //直接执行 Person 的 userAxe()方法。
            p.useAxe();
          }
        }
```

上面的程序执行完成后，将会得到下面的结果：

设置注入

在上述代码中，当主程序调用 Person 的 useAxe()方法时，该方法的方法体内需要使用 Axe 的实例，但程序里没有任何地方将特定的 Person 实例和 Axe 实例耦合在一起。或者说，程序里没有为 Person 实例传入 Axe 的实例，Axe 实例由 Spring 在运行期间动态注入。在 Spring 框架中，Person 实例不仅不需要了解 Axe 实例的具体实现，甚至无须了解 Axe 的创建过程。Person 实例运行到需要 Axe 实例的地方，自然就产生了 Axe 实例，用来供 Person 实例使用。在设置注入中，用户可以省去很多事情，不必关心其具体实现的过程。下面给出一段程序，用来编译和运行该应用的简单脚本。

```
<?xml version="1.0"?>
<!-- 定义编译该项目的基本信息-->
<PROJECT name="spring" default="." basedir=".">
<!-- 定义编译和运行该项目时所需的库文件 -->
<PATH id=classpath>
  <!-- 该路径下存放 spring.jar 和其他第三方类库 -->
  <FILESET dir=..\..\lib>
    <INCLUDE name="*.jar" />
  </FILESET>
  <!-- 同时还需要引用已经编译过的 class 文件-->
  <PATHELEMENT path="." />
</PATH>
<!-- 编译全部的 java 文件-->
<TARGET description="Compile all source code" name="compile">
<!-- 指定编译后的 class 文件的存放位置 -->
<JAVAC debug="true" destdir=".">
  deprecation="false" optimize="false" failonerror="true">
  <!-- 指定需要编译的源文件的存放位置 -->
  <SRC path="." />
  <!-- 指定编译这些 java 文件需要的类库位置-->
  <CLASSPATH refid="classpath" />
</JAVAC>
</TARGET>
<!-- 运行特定的主程序 -->
<TARGET description="run the main class" name="run" depends="compile">
<!-- 指定运行的主程序:lee.BeanTest。-->
<JAVA failonerror="true" fork="yes" classname="lee.BeanTest">
  <!-- 指定运行这些 java 文件需要的类库位置-->
  <CLASSPATH refid="classpath" />
</JAVA>
</TARGET>
</PROJECT>
```

如果需要改写 Axe 的实现类，或者说提供了另一个实现类给 Person 实例使用，此时无须改变 Person 接口和 Chinese 类，只需提供另一个 Axe 的实现，然后对配置文件进行简单的修改即可。

```
//Axe 的另一个实现类 SteelAxe
public class SteelAxeone implements Axe
{
  //默认构造器
  public SteelAxeone()
  {
}
  //实现 Axe 接口的 chop 方法
  public String chop()
```

```
        {
          return "设置注入 1";
        }
      }
```

当写好类后可以修改配置文件，在其中增加如下代码。

```
      <!-- 定义一个 steelAxeone bean-->
      <BEAN class=lee.SteelAxeone id=steelAxeone />
```

该行重新定义了一个 Axe 的实现“SteelAxe”，然后修改 Chinese Bean 的配置，将原来传入 stoneAxe 的地方改为传入 steelAxe。也就是将下面的代码进行修改。

```
      <REF local=""stoneAxeone"/">
```

修改成：

```
      <REF local=""steelAxeone"/">
```

将修改后的程序保存，执行后得到如下结果。

设置注入 1

18.3.3　构造注入

依赖关系除了上面的设置注入外，还有另外一种方式构造注入。所谓构造注入，指通过构造函数来完成依赖关系的设定，而不是通过 Setter 方法。我们可以对前面代码中的 Chinese 类做简单修改，修改后的代码如下。

```
      //Chinese 实现 Person 接口
      public class Chineseone implements Personone
      {
        //面向 Axe 接口编程，而不是具体的实现类
 private Axe axe;
        //默认的构造器
        public Chinese()
        {
      }
        //构造注入所需的带参数的构造器
        public Chinseone(Axe axe)
        {
          this.axe = axe;
        }
        //实现 Person 接口的 useAxe 方法
        public void useAxe()
        {
          System.out.println(axe.chop());
        }
      }
```

通过 Chinese 类里的 setAxe()方法，在构造 Person 实例时 Spring 为 Person 实例注入所依赖的 Axe 实例。构造注入的配置文件也需做简单的修改，修改后的配置文件如下。

```
<!-- 下面是标准的 XML 文件头 -->
<xml version="1.0" encoding="gb2312"?>
<!-- 下面一行定义 Spring 的 XML 配置文件的 dtd -->
  "http://www.springframework.org/dtd/spring-beans.dtd">
<!-- 以上三行对所有的 Spring 配置文件都是相同的 -->
<!-- Spring 配置文件的根元素 -->
<BEANS>
  <!—定义第一个 bean，该 bean 的 id 是 chinese,class 指定该 bean 实例的实现类 -->
  <BEAN class=lee.Chinese id=chinese>
</BEAN>
<!-- 定义 stoneAxe bean -->
<BEAN class=lee.SteelAxe id=steelAxe />
</BEANS>
```

此段代码执行效果与使用 steelAxeone 设置注入时的执行效果完全一样。区别在于：创建 Personone 实例中 Axe 属性的时机不同。设置注入是先创建一个默认的 Bean 实例，然后调用对应的构造方法注入依赖关系。而构造注入则在创建 Bean 实例时，已经完成了依赖关系的设置。

18.4 Spring 的容器

Spring 容器是设计精妙的机器，它的外在表现是通过精细的内部设计来实现的。本节将对 Spring 容器进行解析，从内部探究 Spring 容器的体系结构和运行流程。

18.4.1 Spring 常见的两个接口

Spring 两个重要接口，分别是 BeanFactory 和 ApplicationContext。其中 BeanFactory 是 Spring 核心接口，它为 Spring 提供了一种高级的配置机制，可用所有可能的存储介质保存任意对象的配置信息。对象间通常会相互协作，我们也可以说它们相互间具有依赖性。这些依赖性通过 BeanFactory 的配置数据反映出来。（但某些依赖性从配置数据中是看不到的，比如运行时对象之间的方法调用。）

ApplicationContext 提供了管理和操作对象的基本功能，但大多数需要以编程的方式来处理。ApplicationContext 接口则扩展了 BeanFactory 接口的功能。上下文模型的基础就是 ApplicationContext 接口，该接口位于 SpringContext 命名空间。ApplicationContext 除了具备基本的容器功能外，还能支持更多的应用程序框架，如资源的获取、文字消息的解析、事件的处理。

Spring 最基本的接口是 BeanFactory，它负责配置、创建、管理 Bean。在 BeanFactory 接口中包含以下几个常用方法。

- boolean containsBean(String name)：用来判别容器是否包含 id 为 name 的 Bean 的一种实例。

- Object getBean(String name,Class requiredType)：将容器中的 id 返回为 name，并且类型为 requiredType 的 Bean。
- Object getBean(String name)：用来返回容器中的方法 id 为 name 的 Bean 实例。
- Class getType(String name)：用来返回容器中指定的 Bean 实例的类型。

BeanFactory 有很多实现类，仅使用它提供的容器管理功能是不够的，Spring 还为我们提供了更高级的功能，例如，使用 org.springframework.beans.factory.xml 类。但对于大部分 J2EE 应用而言，推荐使用 ApplicationContext.Application-Context 接口，它是 BeanFactory 的子接口，其常用实现类是 org.springframework.context.support.FileSystemXmlApplicationContext 和 org.springframework.context.support.ClassXml。

在使用 AplicationContext 创建 BeanFactory 实例时，需要提供 Spring 容器的 Bean 的详细配置信息。 Spring 的配置信息一般是采用 XML 文件设置，因此在创建 BeanFactory 实例时，应该提供 XML 配置文件作为参数，XML 配置文件通常使 Resource 对象传入。

18.4.2　使用 ApplicationContext

在大多数情况下，都不会使用 BeanFactory 实例作为 Spring 容器，而是使用 ApplicationConyext 实例作为容器，因此有时把 Spring 容器称为 Spring 的上下文。在前面已经讲解过，applicationContext 是 BeanFactory 的子接口。使用 ApplicationContext 后，可以让系统加载我们的 Bean（例如，在 Servlet 容器初始化 ContextLoaderServlet 时，通过 ContextLoader 类加载 Spring Framework），而不是使用编码方式来加载。ApplicationContext 接口是 context 包的基础，位于 org.springframework.context 包里，提供了 BeanFactory 的所有功能。除此之外， ApplicationContext 为了支持 Framework 的工作方式，还提供了以下的功能。

- 通过 MessageSource 支持语言信息的国际化。
- 对资源（如 URL 和文件系统）的访问。
- ApplicationListener 接口的 Bean 提供了事件传播支持。
- 在复杂的应用环境中提供不同的 context，例如，支持 web 应用的 XmlWeb ApplicationContext 类。

ApplicationContext 包含了 BeanFactrory 的所有功能，因此在使用的时候，建议优先使用 ApplicationContext，除非是在非常特殊的时期，如内存非常关键的应用，才考虑使用 BeanFactory 接口。

ApplicationContext 继承了 MessageSoure 接口，因此它具备国际化功能，下面将简单介绍 ApplicationContext 接口的国际化功能，其介绍如下。

- String getMessage(String code,Object[]args,String default,Local loc)。
- String getMessage(String code,Object[]args,Local loc)。
- String getMessage(MessageSourceResolvable,Local loc)。

下面通过代码来讲解 applicationcontext 国际化的方法，在配置文件中的代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
   <!DOCTYPE beans Public "-//SPRING//DTD BEAN//EN"
 "http://www.springframework.org/dtd/spring-beans.dtd">
```

```
            <beans>
        <bean id ="messageSource" class="org.springframework.context.support.R
esourceBundleMessageSource">
        <property name="basename">
         <value>message</value>
        </property>
        </bean>
        <bean id ="HelloWorld" class="com.jnotnull.HelloWorld">
         <bean>
```

在这里我们定义一个 bean，并且定义了 basename 的属性，属性值为 message。这样我们在建立资源文件的时候文件名就必须是 message 了，扩展名各不相同。下面我们建立 message.properties 文件，在这里要对上面的汉字进行转码，因为有第三方的插件，所以不会产生乱码。再看下面笔者编写的一个测试类。

```
        package com.jnotnull;   //包文件
        import org.springframework.context.ApplicationContext;//引入系统的包
        import org.springframework.context.support.FileSystemXmlApplicationCon
text//插入系统包
        import java.util.Date; //插入数据包
        public class Test {   //建立类
            public static void main(String []args){
                ApplicationContext context = new FileSystemXmlApplicationConte
xt("config.xml");
                        Object[] objects = new Object[]{"HelloWorld",new Date(
)};   //实例化一个项目
                String message= context.getMessage("HelloWorld",objects,Locale
.CHINA);
                System.out.println(message);
            }
        }
```

这样就输出了结果，同时还可以创建另一个资源文件 message_en_US.properties，将如下代码。

```
        String message= context.getMessage("HelloWorld",objects,Locale.CHINA);
```

修改为下面的代码。

```
        String message= context.getMessage("HelloWorld",objects,Locale.US);
```

修改代码后就得到了国际化的处理，整个过程十分简单。

上面的代码不能完全体现国际化的功能。一个国际化的功能，通常会在不同的语言环境下，显示不同的文字，如在中文环境下显示中文，在英文环境下显示英文，如下面的（光盘：源代码/第 18 章/18-3/）代码。

配置文件 bean.xml，其代码如下。

```
<?xml version="1.0" encoding="GBK" ?>
 <!--调用 XML 版本和语言类型 -->
- <project name="hibernte" basedir="." default="">
  <property name="src" value="src" />
  <property name="dest" value="classes" />
- <path id="classpath">
- <fileset dir="..\..\lib">
  <include name="*.jar" />
 <!--装载必要的 Jar 文件 -->
/fileset>
  <pathelement path="${dest}" />
  </path>
- <target name="compile" description="Compile all source code">
  <delete dir="${dest}" />
 <!--使出文件列表 -->
  <mkdir dir="${dest}" />
 <!--建立新的子目录 -->
- <copy todir="${dest}">
- <fileset dir="${src}">
  <exclude name="**/*.java" />
  </fileset>
  </copy>
- <javac destdir="${dest}" debug="true" deprecation="false" optimize=
"false" failonerror="true">
  <src path="${src}" />
  <classpath refid="classpath" />
  </javac>
  </target>
- <target name="run" description="Run the main class" depends=
"compile">
- <java classname="lee.SpringTest" fork="yes" failonerror="true">
  <classpath refid="classpath" />
  </java>
  </target>
  </project>
```

编写一段 Java 代码，其代码如下。

```
package lee;//包文件
import org.springframework.context.ApplicationContext;//插入包
import org.springframework.context.support.ClassPathXmlApplicationContext;
import java.util.*;
//建立一个 springTest 类
public class SpringTest
{
   public static void main(String[] args)throws Exception
   {
```

```
            //实例化 ApplicationContext
            ApplicationContext one = new
                ClassPathXmlApplicationContext("bean.xml");
            //创建参数数组
            String[] a = {"亲爱的朋友"};
            //使用 getMessage 方法获取本地化消息。Locale 的 getDefault 方法
            //返回计算机环境的默认 Locale
            String hello = one.getMessage("hello",a,Locale.getDefault());
            Object[] b = {new Date()};
            String now = one.getMessage("hello",b,Locale.getDefault());
            //打印出两条本地化消息
            System.out.println(hello);
            System.out.println(now);
        }
    }
```

编译后，在中文环境下，会得到下面的结果。

欢迎你，亲爱的朋友

现在时间是 11-4-5，下午 20:42

在英文环境下，会得到下面的结果。

Welcome，亲爱的朋友

now is ：5/4/11 20:42 PM

提示：Spring 国际化支持，其实是建立在 Java 程序国际化的基础上，其核心思路都是将程序中需要实现国际化的信息写入资源文件里，而代码只是使用相应的信息密钥。

18.4.3　ApplicationContext 事件机制

ApplicationContext 中的事件处理是通过 ApplicationEvent 类和 ApplicationListener 接口来实现的。如果在上下文中部署一个实现了 ApplicationListener 接口的 bean，那么每当一个 ApplicationEvent 发布到 ApplicationContext 时，这个 bean 就得到通知。实质上，这是标准的 Observer 设计模式。Spring 提供了如下 3 个标准事件。

- ContextRefreshedEvent：当 ApplicationContext 初始化或刷新时发送的事件。这里的初始化意味着所有的 bean 被装载，singleton 被实例化，以及 ApplicationContext 已就绪可用。
- ContextClosedEvent：当使用 ApplicationContext 的 close()方法结束上下文时发送的事件。这里的结束意味着 singleton bean 被销毁。
- RequestHandledEvent：一个与 Web 相关的事件，告诉所有的 bean 一个 HTTP 请求已经被响应了（也就是在一个请求结束后会发送该事件）。注意，只有在 Spring 中使用了 DispatcherServlet 的 Web 应用才能使用。

下面通过一段代码来演示 Spring 容器的事件机制，在程序中先定义了一个类，其代码（光盘：源代码/第 18 章/181-2）如下。

```
package lee;//包文件
```

```
import org.springframework.context.ApplicationEvent;//插入包
public class EmailEvent extends ApplicationEvent //建立类
{
   //为容器事件类定义无参数的构造器
   String address;//建立地址属性
   String text;//建立文本属性
   public EmailEvent(Object source)
   {
         super(source);
   }
   //定义一个有参数的构造器。
   public EmailEvent(Object source ,
         String address , String text)
   {
         super(source);
         this.address = address;
         this.text = text;
   }
}
```

编写一个类来触发事件，实现代码如下。

```
package lee;//包文件
import org.springframework.context.ApplicationListener;//插入包
import org.springframework.context.ApplicationEvent;//插入包
public class EmailNotifier implements ApplicationListener//插入包
{
   //该方法会在容器发生事件时自动触发
   public void onApplicationEvent(ApplicationEvent evt)
   {
         if (evt instanceof EmailEvent)
         {
               //发送 email 通知...
               EmailEvent emailEvent = (EmailEvent)evt;
               System.out.println("需要发送邮件的接收地址  "
                     + emailEvent.address);
               System.out.println("需要发送邮件的邮件正文  "
                     + emailEvent.text);
         }
         else
         {
               //容器内置事件不作任何处理
               System.out.println("容器本身的事件:   " + evt);
         }
   }
```

```
        }
```

将监听器放在容器中，配置文件如下。

```
        <?xml version="1.0" encoding="GBK" ?>
        - <!--
         指定 Spring 配置文件的 DTD 信息
          -->
          <!DOCTYPE beans (View Source for full doctype...)>
        - <!--
         Spring 配置文件的根元素
          -->
        -  <beans  default-lazy-init="false"  default-autowire="no"  default-
dependency-check="none
         default-merge="false">
        - <!--
         配置监听器
          -->
        <bean  id="emailListListener"  class="lee.EmailNotifier"  lazy-init=
"default" autowire="default"
         dependency-check="default" />
          </beans>
```

当系统创建 Spring 容器后，加载容器会自动触发容器事件，容器事件监听器可以监听到这些事件。除此之外，程序可以调用 ApplicationContex 的 publishEvent()方法来主动触发容器事件，程序如下。

```
        package lee;//包
        import org.springframework.context.ApplicationContext;//插入包
        import org.springframework.context.support.ClassPathXmlApplicationContext;
//插入包
        public class SpringTest{//建立类
           public static void main(String[] args){
              ApplicationContext one = new
                    ClassPathXmlApplicationContext("bean.xml");//实例化配置文件
               //创建一个 ApplicationEvent 对象
               EmailEvent ele = new EmailEvent
                    ("hello","spring_text@163.com","this is a test");//实例化
               //主动触发容器事件
               one.publishEvent(ele);
           }
        }
```

执行上述程序，得到如图 18-10 所示的效果。

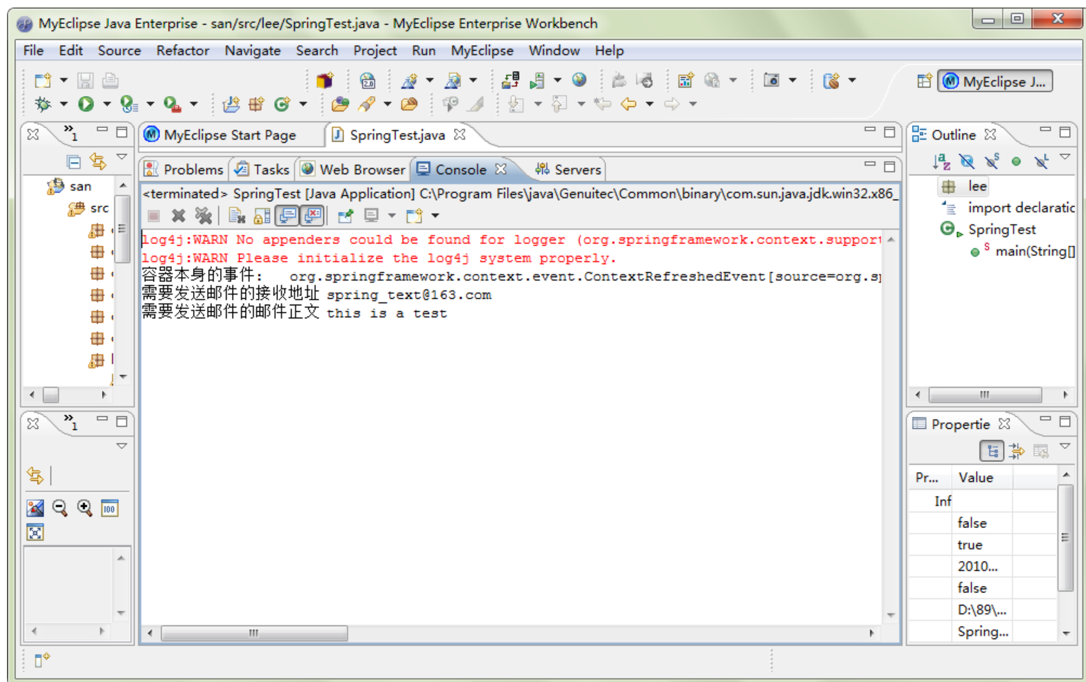


图 18-10　执行结果

18.5　Bean 对 Spring 的意义深远

Spring 框架绝大部分都放在 Bean 容器中，包括管理容器中 Bean 的生命周期，使用 Bean 继承等特殊功能。通过这些深入的管理，可以更好地使用这些 Java 组件。

18.5.1　Bean 的基本定义

<beans…/>元素是 Spring 配置文件的根元素，<bean…/>是 Bean 元素的子元素。一个程序可以包含多个<bean…/>元素，每个 Bean 对应 Spring 的一个实例，下面通过定义一个用户持久化来讲解这些元素的作用，其代码如下。

```
<bean id="userDao" class="com.dev.spring.simple.MemoryUserDao"/>
```

这是一个最简单的 Bean 定义。它类似于调用了语句 MemoryUserDao userDao = new MemoryUserDao()。id 属性必须是一个有效的 XML ID，这意味着它在整个 XML 文档中必须唯一。它是一个 Bean 的“终身代号”。同时也可以用 name 属性为 Bean 定义一个或多个别名（用逗号或空格分开多个别名）。name 属性允许出现任意非法的 XML 字母，例如，下面的代码。

```
<bean id="userDao" name="userDao*_1, userDao*_2"
class="com.dev.spring.simple.MemoryUserDao"/>。
```

class 属性定义了这个 Bean 的全限定类名（包名＋类名）。Spring 能管理几乎所有的 Java

类。一般情况下，这个 Java 类会有一个默认的构造函数，用 set()方法设置依赖的属性。

Bean 元素除了上面的两个属性之外，还有很多其他属性。下面通过一段代码来讲解 Bean 元素的其他属性，其介绍如下。

```
<bean
    id="beanId"（1）//id 号
    name="beanName"（2）//姓名
    class="beanClass"（3）//班级
    parent="parentBean"（4）
    abstract="true | false"（5）//抽象
    singleton="true | false"（6）
    lazy-init="true | false | default"（7）
    byName | byType | constructor | autodetect | default"（8）
    dependency-check = "none | objects | simple | all | default"（9）
    depends-on="dependsOnBean"（10）
    init-method="method"（11）
    destroy-method="method"（12）
    factory-method="method"（13）
  factory-bean="bean">（14）
</bean>
```

下面对上面这段代码进行讲解，其介绍如下。

- ❑ id：Bean 的唯一标识。它必须是合法的 XML ID，在整个 XML 文档中唯一。
- ❑ name：用来为 id 创建一个或多个别名。它可以是任意的字母组合。多个别名之间用逗号或空格分开。
- ❑ class: 用来定义类的全限定名（包名＋类名）。只有子类 Bean 不用定义该属性。
- ❑ parent: 子类 Bean 定义它所引用的父类 Bean。这时前面的 class 属性失效。子类 Bean 会继承父类 Bean 的所有属性，子类 Bean 也可以覆盖父类 Bean 的属性。注意，子类 Bean 和父类 Bean 是同一个 Java 类。
- ❑ abstract（默认为“false”）：用来定义 Bean 是否为抽象 Bean。它表示这个 Bean 将不会被实例化，一般用于父类 Bean，因为父类 Bean 主要是供子类 Bean 继承使用。
- ❑ singleton（默认为“true”）：定义 Bean 是否是 Singleton（单例）。如果设为“true”，则在 BeanFactory 作用范围内，只维护 Bean 的一个实例；如果设为“flase”，Bean 将是 Prototype（原型）状态，BeanFactory 将为每个 Bean 请求创建一个新的 Bean 实例。
- ❑ lazy-init（默认为“default”）：用来定义这个 Bean 是否实现初始化。如果为“true”，它将在 BeanFactory 启动时初始化所有的 Singleton Bean；反之，如果为“false”，它只在 Bean 请求时才开始创建 Singleton Bean。
- ❑ autowire（自动装配，默认为“default”）：它定义了 Bean 的如下自动装载方式。
  - ●“no”：不使用自动装配功能。
  - ●“byName”：通过 Bean 的属性名实现自动装配。
  - ●“byType”：通过 Bean 的类型实现自动装配。
  - ●“constructor”：类似于 byType，但它是用于构造函数的参数的自动组装。

  - "autodetect"：通过 Bean 类的反省机制（introspection）决定是使用"constructor"还是使用"byType"。
- ❑ dependency-check（依赖检查，默认为"default"）：它用来确保 Bean 组件通过 JavaBean 描述的所有依赖关系都得到满足。在与自动装配功能一起使用时特别有用，具体取值如下。
  - none：不进行依赖检查。
  - objects：只做对象间依赖的检查。
  - simple：只做原始类型和 String 类型依赖的检查。
  - all：对所有类型的依赖进行检查。它包括了前面的 objects 和 simple。
- ❑ depends-on（依赖对象）：Bean 在初始化时依赖的对象，这个对象会在这个 Bean 初始化之前创建。
- ❑ init-method：用来定义 Bean 的初始化方法，它会在 Bean 组装之后调用。它必须是一个无参数的方法。
- ❑ destroy-method：用来定义 Bean 的销毁方法，它在 BeanFactory 关闭时调用。同样，它也必须是一个无参数的方法。它只能应用于 singleton Bean。
- ❑ factory-method：定义创建该 Bean 对象的工厂方法。它用于下面的"factory-bean"，表示这个 Bean 是通过工厂方法创建的。此时"class"属性失效。
- ❑ factory-bean：定义创建该 Bean 对象的工厂类，如果使用了"factory-bean"则"class"属性失效。

18.5.2　容器中 Bean 的作用域

在 Spring 框架中，不但可以完成 Bean 实例的实例化，还可以为 Bean 指定特定的作用域，Spring 主要有 5 种作用域，其介绍如下。

- ❑ singleton：单例模式，使用 singleton 定义的 Bean 将只有一个实例。
- ❑ prototype：原型模式，通过容器 getBean 方法获取 Bean 时，都将产生一个新的 Bean 实例。
- ❑ globalSession：全局 HTTP Session。
- ❑ request：每次 HTTP 请求，都将产生一个新实例。
- ❑ Session：每次 HTTP Session，都将产生一个新实例。

提示：

Spring 默认使用 singleton 作用域。singleton 作用域的 Bean 一旦创建后即可重复使用。因为在创建和销毁 prototype 作用域的 Bean 时，代价会较大，所以应尽量避免将 Bean 定义为 prototype 作用域。

1．singleton 作用域

当一个 Bean 的作用域设置为 singleton 时，Spring IOC 容器中只会存在一个共享的 Bean 实例，并且所有对 Bean 的请求，只要 id 与该 bean 定义相匹配，则只会返回 Bean 的同一实例。换言之，当把一个 Bean 定义设置为 singleton 作用域时，Spring IOC 容器只会创建该 bean 定义的唯一实例。这个单一实例会被存储到单例缓存（singleton cache）中，并且所有

针对该 Bean 的后续请求和引用都将返回被缓存的对象实例，这里要注意的是 singleton 作用域和 GOF 设计模式中的单例是完全不同的，单例设计模式表示一个 ClassLoader 中只有一个 class 存在，而这里的 singleton 则表示一个容器对应一个 Bean，也就是说当一个 Bean 被标识为 singleton 时候，Spring 的 IOC 容器中只会存在一个 Bean，下面通过一段代码进行讲解，其代码如下。

```
<bean id="role" class="spring.chapter2.maryGame.Role" scope="singleton"/>
```

或者编写成如下代码。

```
<bean id="role" class="spring.chapter2.maryGame.Role" singleton="true"/>
```

2．prototype 作用域

使用 prototype 作用域部署的 Bean，每一次请求（将其注入到另一个 Bean 中，或者以程序的方式调用容器的 getBean()方法）都会产生一个新的 Bean 实例，相当于一个 new 的操作。对于 prototype 作用域的 Bean，有一点非常重要，那就是 Spring 不能对一个 prototype bean 的整个生命周期负责，容器在初始化、配置、装饰或装配完一个 prototype 实例后，将它交给客户端，随后就对该 prototype 实例不闻不问了。不管何种作用域，容器都会调用所有对象的初始化生命周期回调方法，而对 prototype 而言，任何配置好的析构生命周期回调方法都将不会被调用。清除 prototype 作用域的对象并释放任何 prototype bean 所持有的昂贵资源，都是客户端代码的职责（让 Spring 容器释放被 singleton 作用域 bean 占用资源的一种可行方式是，通过使用 Bean 的后置处理器，该处理器持有要被清除的 Bean 的引用），例如下面的代码。

```
bean id="role" class="spring.chapter2.maryGame.Role" scope="prototype"/>
```

或者编写成如下代码。

```
<beanid="role" class="spring.chapter2.maryGame.Role" singleton="false"/>
```

3．request 作用域

request 表示针对每一次 HTTP 请求都会产生一个新的 Bean，同时该 Bean 仅在当前 HTTP request 内有效，配置实例 request、session 和 global session 使用的时候，首先要在初始化 Web 的 web.xml 中做进行配置。如果使用的是 Servlet 2.4 及以上的 Web 容器，那么仅需要在 Web 应用的 XML 声明文件 web.xml 中增加下述 ContextListener 即可。

```
<web-app>
  ...
  <listener>
<listener-class>org.springframework.web.context.request.
RequestContextListener </listener-class>
  </listener>
  ...
</web-app>
```

如果是 Servlet 2.4 以前的 Web 容器，那么要使用一个 javax.servlet.Filter 来实现，其代码如下。

```
        <web-app>
         ..
         <filter>
        <filter-name>requestContextFilter</filter-name>
             <filter-class>org.springframework.web.filter.RequestContextFilter
</filter-class>
         </filter>
         <filter-mapping>
            <filter-name>requestContextFilter</filter-name>
            <url-pattern>/*</url-pattern>
         </filter-mapping>
          ...
        </web-app>
```

接下来配置 Bean 的作用域，其代码如下。

```
        bean id="role" class="spring.chapter2.maryGame.Role" scope="request"/>
```

4．Session 作用域

Session 作用域表示针对每一次 HTTP 请求都会产生一个新的 Bean，同时该 Bean 仅在当前 HTTP Session 内有效，其代码如下。

```
        <bean id="role" class="spring.chapter2.maryGame.Role" scope="session"/>
```

5．Global Session 作用域

Global Session 作用域类似于标准的 HTTP Session 作用域，不过它仅在基于 Portlet 的 Web 应用中才有意义。Portlet 规范定义了全局 Session 的概念，它被所有构成某个 Portlet Web 应用的各种不同的 Portlet 所共享。在 Global Session 作用域中定义的 Bean 被限定于全局 Portlet Session 的生命周期范围内。如果在 Web 中使用 Global Session 作用域来标识 Bean，那么 Web 会自动当成 Session 类型来使用，其代码如下。

```
        <bean id="role" class="spring.chapter2.maryGame.Role"scope="global session"/>
```

6．自定义 Bean 装配作用域

在 Spring 2.0 中作用域是可以任意扩展的，用户可以自定义作用域，甚至也可以重新定义已有的作用域（但是不能覆盖 singleton 和 prototype）。Spring 的作用域由接口 org.springframework.beans.factory.config.Scope 来定义，在自定义作用域时只要实现该接口即可，其代码如下。

```
        Public class MyScopeone implements Scope//定义一个类实现一个接口
        {
             privatefinal ThreadLocal threadScope = new ThreadLocal()//实例化
        {
```

```
                  protected Object initialValue()
        {
                    returnnew HashMap();//返回
                  }
        }
             public Object get(String name, ObjectFactory objectFactory)
        {
                Map scope = (Map) threadScope.get();
                Object object = scope.get(name);
                if(object==null) {
                  object = objectFactory.getObject();
                  scope.put(name, object);
                }
                return object;
             }
             public Object remove(String name) {//移除方法
                Map scope = (Map) threadScope.get();
                return scope.remove(name);
             }
              publicvoid  registerDestructionCallback(String  name,  Runnable
callback)
        {
             }
            public String getConversationId()
        {
                    returnnull;
            }
          }
```

下面通过一个实例来讲解上面 6 个作用域的用法，目的是让读者更深入地理解 6 个作用的意义，其代码（光盘：源代码/第 18 章/18-5/）的实现流程如下。

1）配置文件代码如下。

```
          <?xml version="1.0" encoding="GBK" ?>
        - <!--
         指定 Spring 配置文件的 DTD 信息
          -->
          <!DOCTYPE beans (View Source for full doctype...)>
        - <!--
         Spring 配置文件的根元素
          -->
        -  <beans  default-lazy-init="false"  default-autowire="no"  default-
dependency-check="none"
         default-merge="false">
        - <!--
         配置一个 singleton Bean 实例
```

```
        -->
         <bean  id="p1"  class="lee.Person"  lazy-init="default"  autowire=
"default" dependency-check="default" />
      - <!--
       配置一个 prototype Bean 实例
        -->
         <bean  id="p2"  class="lee.Person"  scope="prototype"  lazy-init=
"default" autowire="default"
       dependency-check="default" />
        </beans>
```

2）编写一个简单的类 Person，其代码如下。

```
package lee;
public class Person
{
   private int age;
}
```

3）建立一个类为 BeanTest 类，其代码如下。

```
package lee;
import org.springframework.beans.factory.BeanFactory;
import org.springframework.beans.factory.xml.XmlBeanFactory;
import java.io.*;
import org.springframework.core.io.InputStreamResource;
import org.springframework.core.io.ClassPathResource;
public class BeanTest
{
   public static void main(String[] args)throws Exception
   {
         //以加载路径下的 beans.xml 文件创建 Resource 对象
         ClassPathResource res =
              new ClassPathResource("beans.xml");
         //以 Resource 对象为参数，创建 BeanFactory 实例。
         XmlBeanFactory beanFactory = new XmlBeanFactory(res);
         //判断两次请求 singleton 作用域的 Bean 实例是否相等
         System.out.println(beanFactory.getBean("p1")
              == beanFactory.getBean("p1"));
         //判断两次请求 prototype 作用域的 Bean 实例是否相等
         System.out.println(beanFactory.getBean("p2")
              == beanFactory.getBean("p2"));
   }
}
```

运行上述代码，得到如图 18-11 所示的结果。

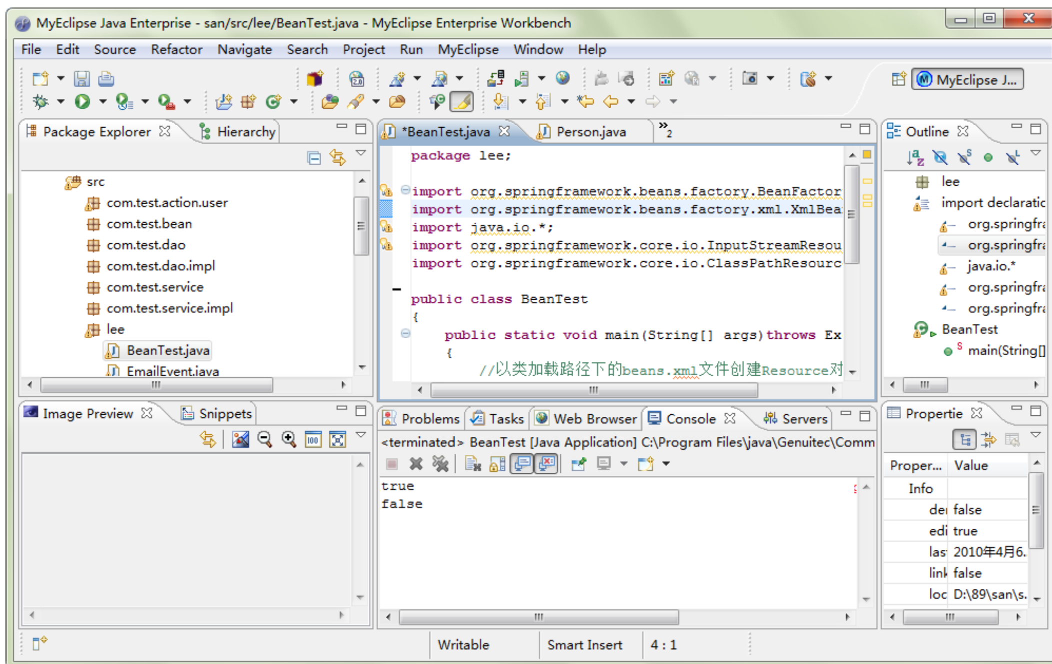


图 18-11　Bean 的作用域

18.5.3　Spring 的 Bean 和 JavaBean

容器对 Bean 没有特殊要求，不需要 Bean 像标准的 JavaBean 那样，必须为每个属性提供对应的 getter()和 setter()方法。Spring 中的 Bean 是 Java 实例、Java 组件，而传统的 Java 应用的 JavaBean 通常作为 DTO，用来封装对象、传递对象。当然 Bean 比 JavaBean 要复杂一些，功能要更强大一点。当然，传统的 JavaBean 也可以作为 Spring 的 Bean，从而接受 Spring 管理。

下面通过一段代码演示 Spring 中 Bean 和 JavaBean 的用法，其代码（光盘：源代码/第 18 章/18-6/）的实现流程如下。

1）建立一个配置文件，其代码如下。

```
<?xml version="1.0" encoding="GBK" ?>
- <!--
 指定 Spring 配置文件的 DTD 信息
-->
 <!DOCTYPE beans (View Source for full doctype...)>
- <!--
 Spring 配置文件的根元素
-->
-  <beans  default-lazy-init="false"  default-autowire="no"  default-dependency-check="none"
 default-merge="false">
- <!--
```

```
        定义数据源 Bean，使用 C3P0 数据源实现
          -->
        - <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"
destroy-method="close" lazy-init="default" autowire="default"dependency-check=
"default">
        - <!--
          指定连接数据库的驱动
          -->
          <property name="driverClass" value="com.mysql.jdbc.Driver" />
        - <!--
          指定连接数据库的 URL
          -->
          <property name="jdbcUrl" value="jdbc:mysql://localhost/javaee" />
        - <!--
          指定连接数据库的用户名
          -->
          <property name="user" value="root" />
        - <!--
          指定连接数据库的密码
          -->
          <property name="password" value="32147" />
        - <!--
          指定连接数据库连接池的最大连接数
          -->
          <property name="maxPoolSize" value="40" />
        - <!--
          指定连接数据库连接池的最小连接数
          -->
          <property name="minPoolSize" value="1" />
        - <!--
          指定连接数据库连接池的初始化连接数
          -->
          <property name="initialPoolSize" value="1" />
        - <!--
          指定连接数据库连接池的连接的最大空闲时间
          -->
          <property name="maxIdleTime" value="20" />
          </bean>
          </beans>
```

2）建立一个数据库 javaee，其 SQL 代码如下。

```
drop database if exists javaee;
create database javaee;
use javaee;
create table newsinf
(
```

```
news_id int auto_increment primary key,
news_title varchar(255));
```

3）编写一段测试代码，其代码如下。

```
package lee;
import org.springframework.beans.factory.BeanFactory;
import org.springframework.beans.factory.xml.XmlBeanFactory;
import java.io.*;
import org.springframework.core.io.InputStreamResource;
import org.springframework.core.io.ClassPathResource;
public class BeanTest
{
   public static void main(String[] args)throws Exception
   {
         //以加载路径下的beans.xml文件创建Resource对象
         ClassPathResource res =
              new ClassPathResource("beans.xml");
         //以Resource对象为参数，创建BeanFactory实例。
         XmlBeanFactory beanFactory = new XmlBeanFactory(res);
         //判断两次请求singleton作用域的Bean实例是否相等
         System.out.println(beanFactory.getBean("p1")
              == beanFactory.getBean("p1"));
         //判断两次请求prototype作用域的Bean实例是否相等
         System.out.println(beanFactory.getBean("p2")
              == beanFactory.getBean("p2"));
   }
}
```

18.6 疑难问题解析

在本章的内容中，详细讲解了 Spring 框架的基本知识。本节中，将对本章中比较难理解的问题进行讲解。

读者疑问：请问 Spring 在 Java 项目中的作用是什么？

解答：Spring 是一个框架，它的功能主要是依靠 Bean 实现，在实际的应用中，它的功能有限，而且有难以克服的缺陷，所以常常会与其他框架融合使用，如与 Hibernat 框架整合使用。Spring 是一个开源框架，是为了解决企业应用程序开发复杂性而创建的。框架的主要优势之一就是其分层架构，分层架构允许用户选择使用哪一个组件，同时为 J2EE 应用程序开发提供集成的框架。

读者疑问：Spring 框架有着许多复杂的技术，该如何深入地学习 Spring 框架呢？

解答：每个框架都大致相同，只要明白 Spring 框架是做什么的，其他的都好办，它也是 Java 代码，用户要想了解功能，只需要看帮助文档即可，每一个类和接口都有相应的解释，不

过用户只需要看常用的即可，不必一一记住。



职场点拨——同事间的互补

在动物界里有些动物之间存在一种奇妙的友好关系，它们互惠互利，相得益彰。其实在人类的现实生活中的互补现象也特别多，尤其是在职场中，这种情况更是屡见不鲜。

（1）案例

A 和 B 是一对工作上的好朋友，她们两个人供职于深圳 XX 广告设计公司，A 负责文案策划，B 负责图片设计制作。刚开始的时候，她们各自负责不同客户的广告设计，不久设计总监就发现她们设计作品的思维和风格有明显的缺陷。B 在绘图能力和电脑操作能力方面比较突出，但是创意方面略显平常；而 A 刚好相反，创意和整体策划都不错，但在绘图方面的表现力始终不尽如人意。

最初她们各自设计的图稿修改了很多次也不能让客户满意，后来设计总监无意中在对两个人的设计进行比较后发现两者居然有互补的倾向。于是，试着让 A 和 B 对同一个客户资料相互沟通，并且合作完成同一个产品的设计方案。

两个人在统一了大体方向后，由 A 负责整个广告方面的文案和策划，由 B 进行绘图方式的表达，这样设计出来的作品结合了两个人的优势，创意独特，让人耳目一新，客户几乎没改动就通过了。

从此以后，她们之间就形成了一种特别的工作关系。在不断地合作过程中越来越默契，两个人因为出色的工作表现成了公司的知名设计组合，同时也为公司赢得了越来越多的客户。

（2）分析

其实像 A 和 B 这种因工作之间能力互补而双赢的现象在职场里比比皆是。互补在职场上意味着一个工作场所里大家互惠互利的相处方式和精诚合作的工作态度。

当今社会的发展分工越来越细致，没有谁是万能的，可以把所有的工作都做到得心应手，我们常常需要在职业中和别人进行沟通和交流，切磋和学习，“共生现象”可以让我们在一个职场里各尽其能，互相合作，以达到把工作做到最佳效果的目的。

在职场里建立一个好的“共生”环境很不容易，很多“共生”都是大家在工作场所和交往里慢慢建立起来的，这种现象也可能由共生对象间产生了矛盾或是一方的职业变更而瓦解。

尺有所短，寸有所长，一个有着凝聚力的工作场所就是一个成功的共生环境。在这个环境里既保持了个体的优势又调动各同事对工作的积极性。在这样的场所里，每一个人都会因为自己能适当地发挥才能而觉得受到了尊重。作为“共生”关系里的一员，对自己的工作充满热情，与同事互为友朋，这才是最根本的“共生”之道。

第 19 章　Hibernate 详解

Hibernate 是一个开放源代码的对象关系映射框架，能够对 JDBC 进行轻量级的对象封装。Hibernate 可以应用于任何使用 JDBC 的场合，既可以在 Java 的客户端程序使用，也可以在 Servlet/JSP 的 Web 应用中使用。并且最具革命意义的是，Hibernate 可以在应用 EJB 的 J2EE 架构中取代 CMP，完成数据持久化的重任。在本章的内容中，将详细讲解 Hibernate 框架的基本知识。通过本章能学到如下知识。

- Hibernate 的下载和安装。
- Hibernate 的基本映射。
- 数据库对象映射。
- 常用接口。
- 职场点拨——谈升职。

2012 年 XX 月 XX 日，多云

今天有一个新的任务，用 Linux+Hibernate 开发一个采购系统，而各个项目任务排得都比较紧，不能再抽调其他项目经理参与这个项目了。急需派一个软件工程师来独立完成这个任务，当然，这个任务最终会成为一个独立项目，进行二次开发。



一问一答

小菜：“我们公司有一个 Linux+Hibernate 项目，需要项目经理独立完成！但是所有项目经理都有任务在身，我想试试！”

Wisdom：“嗯，这显然是一个有升职机会的任务，因为涉及了 Linux+Hibernate。只要你顺利完成了，肯定会升职的！”

小菜：“真的吗？”

Wisdom：“试想作为同级别的同事，Java 的技术水平本身都差不多，你怎样能从技术层面脱颖而出呢？只有从深度和广度上考虑，在深度上 Linux+Hibernate 就是一个好的选择。传统编程语言你和同事们都差不多，但是你还略了解 Linux+Hibernate 的知识，所以肯定会在领导面前发光的。在本章的最后，我简单谈了升职的问题，你可以参考一下！”

19.1　认识 Hibernate

Java 是一门面向对象语言，使用的数据库一直是关系数据库。从某个方面说，数据库没有跟上编程语言的发展，没有跟上面向对象的时代。如果 Java 要操作数据库，只能通过

JDBC 编程来实现。自从有了 Hibernate 后，在用 Java 编程时，可以把数据库看做一个对象来进行操作，这样可以减少开发人员使用 SQL 和 JDBC 处理数据的时间。

19.1.1　Hibernate 常用接口介绍

Hibernate 一共有 5 个核心接口，分别为 Session、SessionFactory、Transaction、Query 和 Configuration。这 5 个核心接口功能强大，能在任何开发中用到，它们之间的关系如图 19-1 所示。

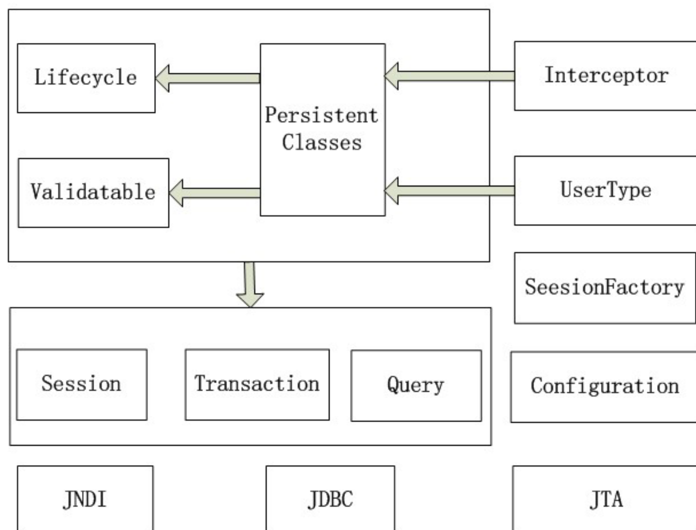


图 19-1　Hibernate 常用接口

通过这些接口，不仅可以对持久化对象进行存取，还能够进行事务控制。上述接口的具体说明如下：

（1）Session 接口

负责执行被持久化对象的 CRUD 操作（CRUD 的任务是完成与数据库的交流，包含了很多常见的 SQL 语句。）。但需要注意的是 Session 对象是非线程安全的。同时，Hibernate 的 session 不同于 JSP 应用中的 HttpSession。这里当使用 session 这个术语时，其实指的是 Hibernate 中的 session，而以后会将 HttpSesion 对象称为用户 session。

（2）SessionFactory 接口

负责初始化 Hibernate。它充当数据存储源的代理，并负责创建 Session 对象。这里用到了工厂模式。需要注意的是 SessionFactory 并不是轻量级的，因为一般情况下，一个项目通常只需要一个 SessionFactory 就够，当需要操作多个数据库时，可以为每个数据库指定一个 SessionFactory。

（3）Configuration 接口

负责配置并启动 Hibernate，创建 SessionFactory 对象。在 Hibernate 的启动过程中，Configuration 类的实例首先定位映射文档位置、读取配置，然后创建 SessionFactory 对象。

（4）Transaction 接口

负责事务相关的操作。它是可选的，开发人员也可以设计编写自己的底层事务处理代码。Query 和 Criteria 接口负责执行各种数据库查询。

19.1.2 Hibernate 下载和安装

在使用 Hibernate 框架之前，需要先进行下载并安装。接下来将详细讲解如何下载和安装 Hibernate，其具体操作流程如下。

1）打开浏览器，在浏览器中输入 Hibernate 的官方网站 http://www.hibernate.org，在打开的官方网站页面单击“Download”超级链接，如图 19-2 所示。

2）拖动滑动条，在“Hibernate Core”选项中单击“Download”超级链接进入下载页面，如图 19-3 所示。

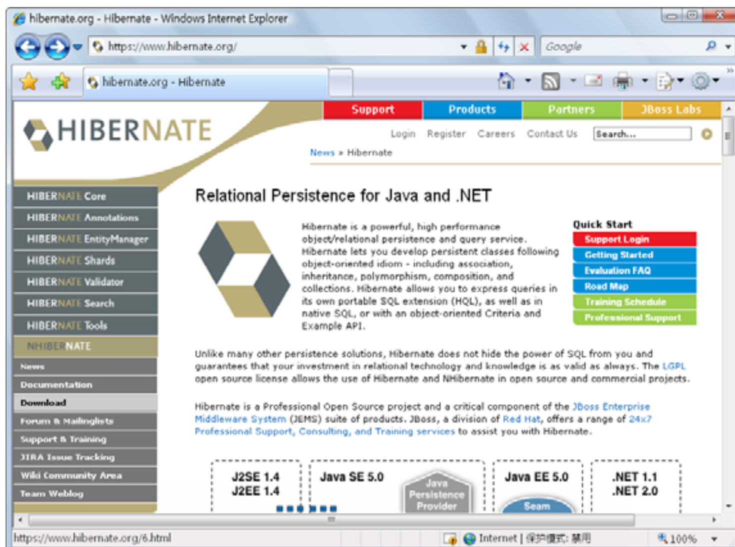


图 19-2 Hibernate 官方首页

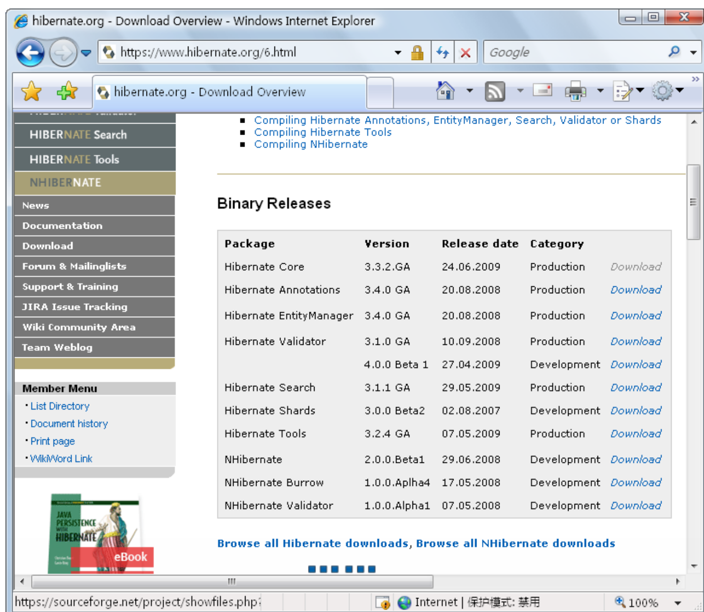


图 19-3 下载

3）选择好下载版本后，需要选择对应的操作系统，如果是 Windows 操作系统则需要下载 zip 压缩包，如果是 Linux 系统则需要下载 tar 包。在此下载 zip 包，如图 19-4 所示。

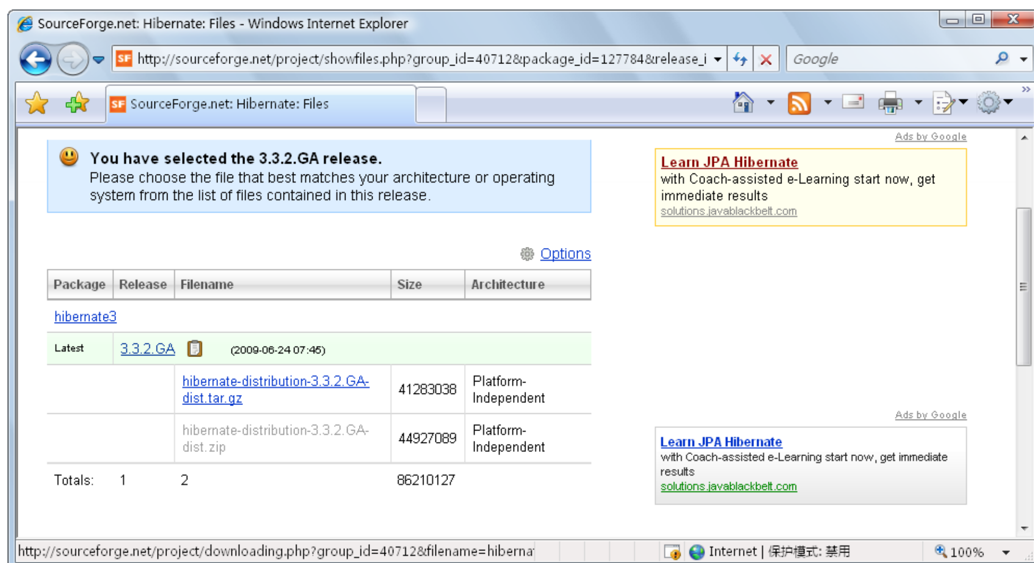


图 19-4　下载压缩包

4）解压下载的压缩包，得到一个名为“Hibernate”的文件夹，如图 19-5 所示。

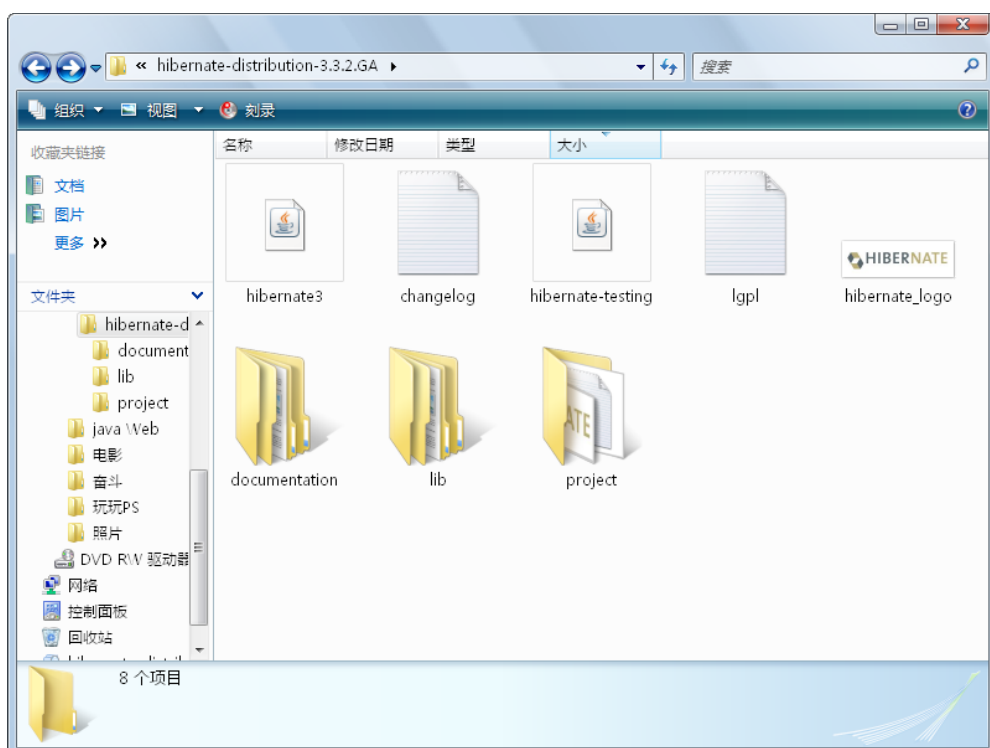


图 19-5　文件夹的文件

5）启动 MyEclipse，将 Hibernate 文件夹的所有 Jar 文件复制到项目中的“lib”文件夹。然后设置新建的 MyEclipse 项目支持 Hibernate。配置完成后的效果如图 19-6 所示。

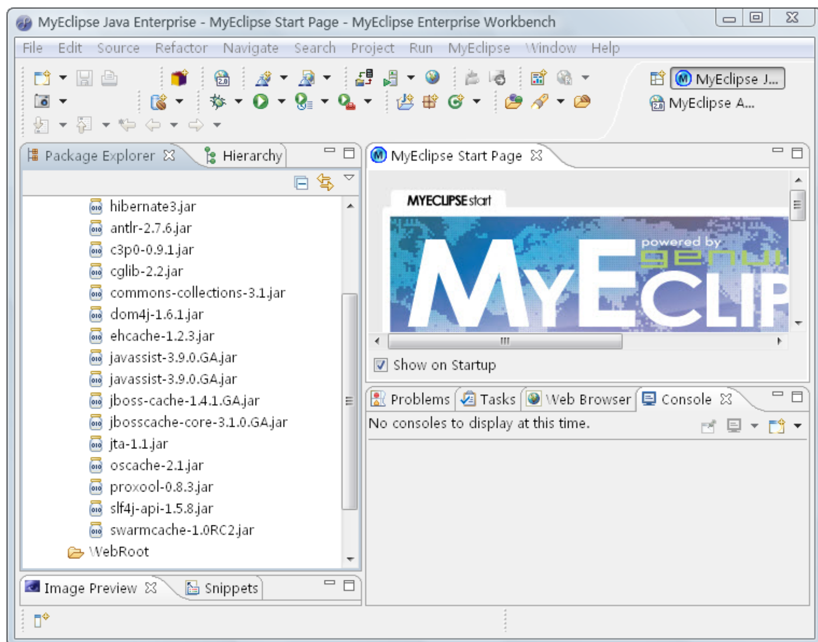


图 19-6　加载支持 Hibernate 的 Jar 文件

6）寻找支持数据的 JDBC 驱动，假设是 MySQL 数据库，则需要将 MySQL 的 JDBC 驱动加载在项目中，如图 19-7 所示。

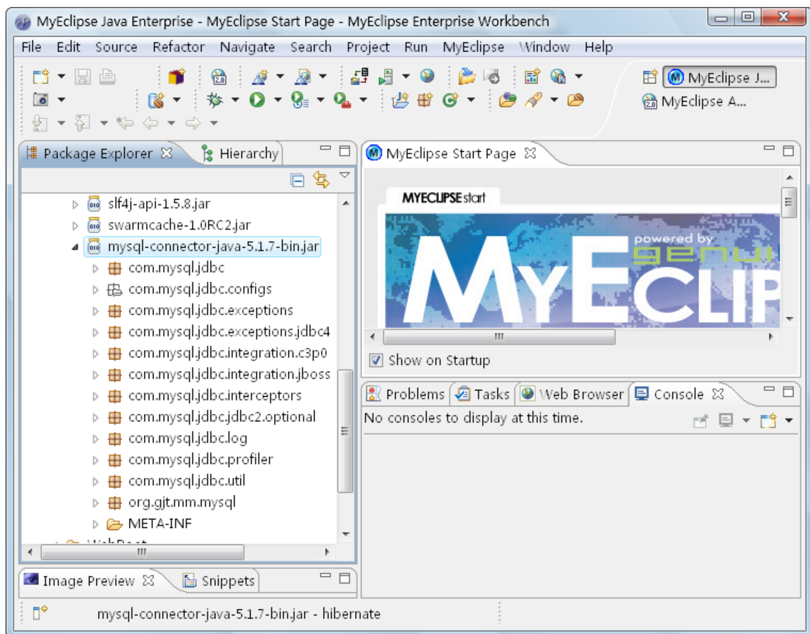


图 19-7　加载数据库的驱动

7）选择“Windows/Open Perspective/Other”命令，如图 19-8 所示。

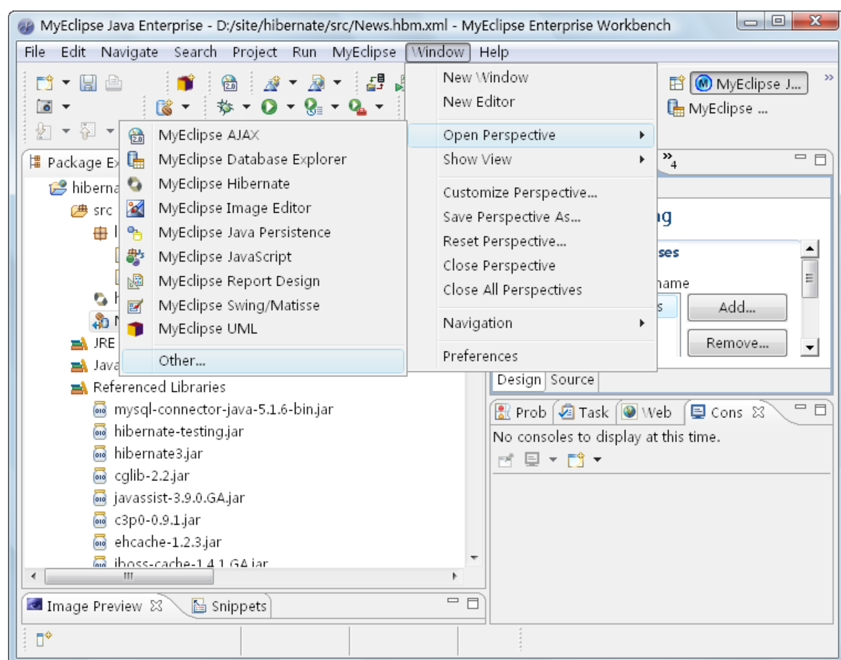


图 19-8　选择命令

8）打开“Open Perspective”对话框，然后选择“MyEclipse Database Explorer”选项，单击“OK”按钮。如图 19-9 所示。

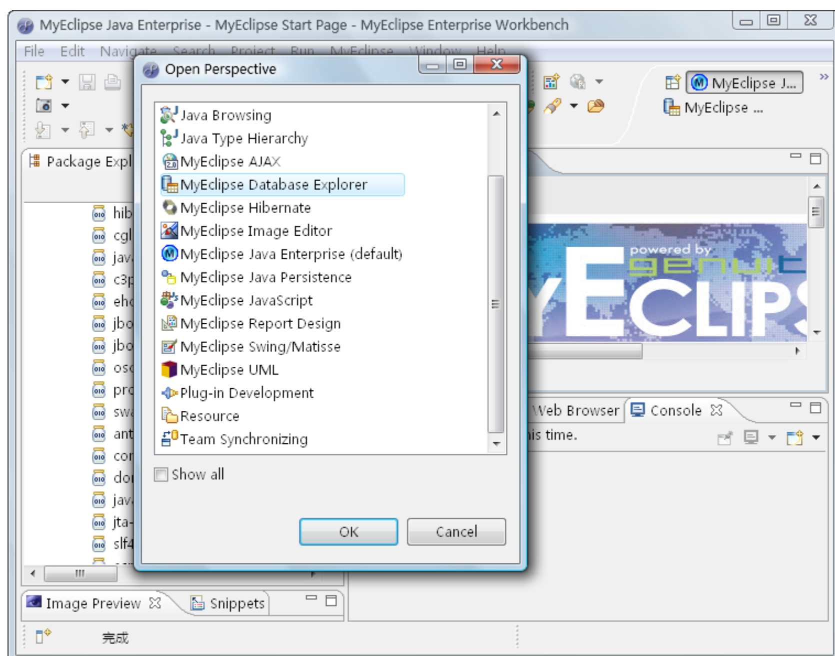


图 19-9　切换视图

9）在打开的视图中的左边单击鼠标右键，在弹出的快捷菜单中选择“new”命令，在打开的“Datebases Driver”窗口中设置数据库信息，设置完成后单击“Finish”按钮。如图 19-10 所示。

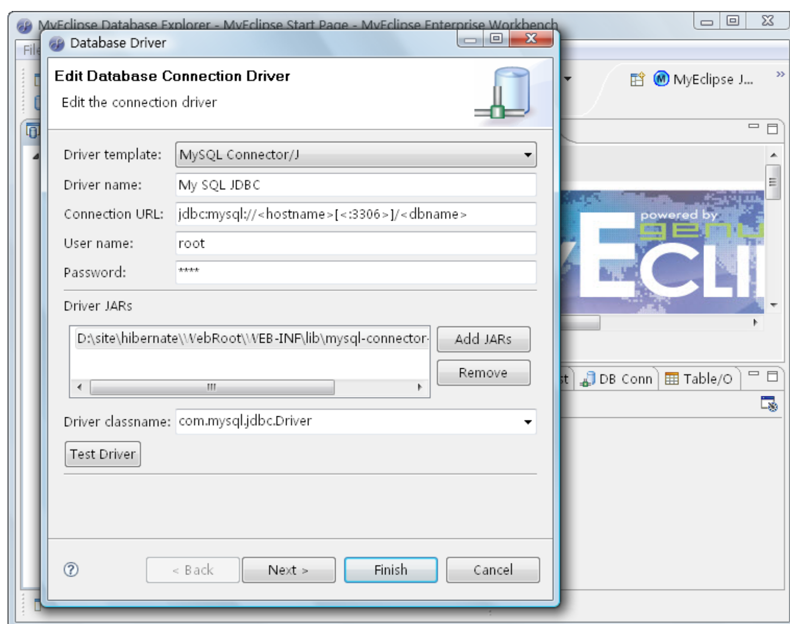


图 19-10　输入设置数据库信息

10）连接成功后，鼠标右键单击此项目，在弹出的菜单中选择“Open another connection”命令后会打开数据库的信息，如图 19-11 所示。

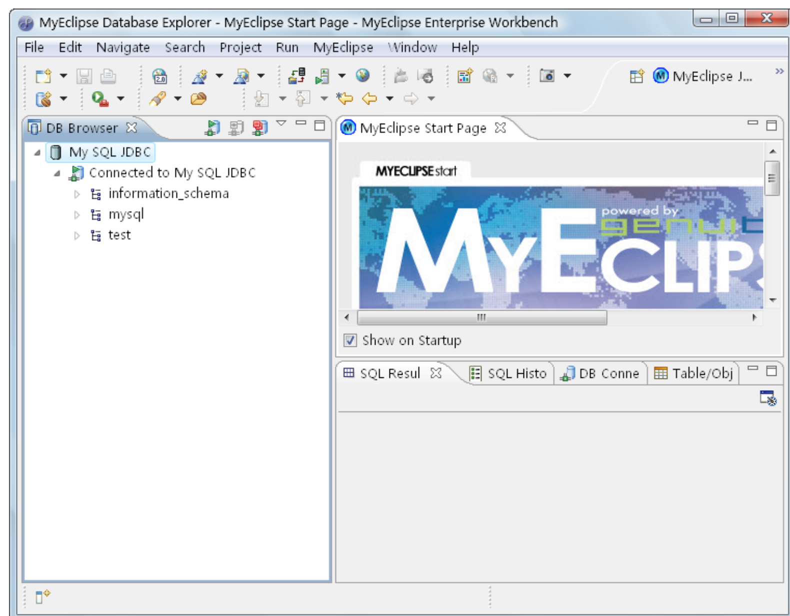


图 19-11　查看到数据库信息

提示：本书以 MySQL 数据库为例进行讲解，读者在连接之前`一定要启动 MySQL 服务器。

11）切换到原来的视图，新建一个 hibernate.cfg.xml 文件，界面如图 19-12 所示。

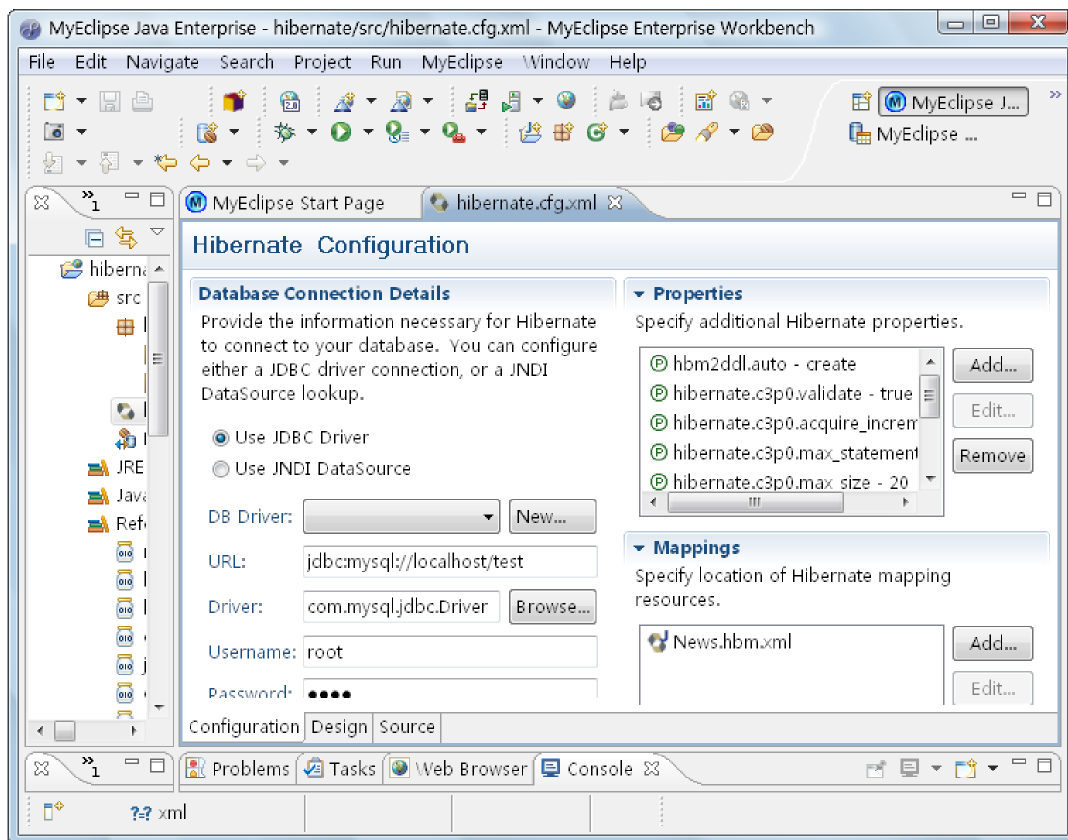


图 19-12　可视化界面

很多程序员不习惯上述 IDE 的可视化操作，一般都是亲自编写实现代码，具体代码如下。

```
<?xml version="1.0" encoding="GBK"?>
<!-- 指定 Hibernate 配置文件的 DTD 信息 -->
<!DOCTYPE hibernate-configuration PUBLIC
   "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
   "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<!-- hibernate- configuration 是连接配置文件的根元素 -->
<hibernate-configuration>
```

```
        <session-factory>
            <!-- 指定连接数据库所用的驱动 -->
            <property name="connection.driver_class">com.mysql.jdbc.Driver
</property>
            <!-- 指定连接数据库的 url，hibernate 连接的数据库名 -->
            <property    name="connection.url">jdbc:mysql://localhost/test
</property>
            <!-- 指定连接数据库的用户名 -->
            <property name="connection.username">root</property>
            <!-- 指定连接数据库的密码 -->
            <property name="connection.password">1234</property>
            <!-- 指定连接池里最大连接数 -->
            <property name="hibernate.c3p0.max_size">20</property>
            <!-- 指定连接池里最小连接数 -->

            <property name="hibernate.c3p0.min_size">1</property>
            <!-- 指定连接池里连接的超时间 -->
            <property name="hibernate.c3p0.timeout">5000</property>
            <!-- 指定连接池里最大缓存多少个 Statement 对象 -->
            <property name="hibernate.c3p0.max_statements">100</property>
            <property name="hibernate.c3p0.idle_test_period">3000 </property>
            <property name="hibernate.c3p0.acquire_increment">2</property>
            <property name="hibernate.c3p0.validate">true</property>
            <!-- 指定数据库方言 -->
            <property  name="dialect">org.hibernate.dialect.MySQLInnoDBDialect
</property>
            <!-- 根据需要自动创建数据库 -->
            <property name="hbm2ddl.auto">create</property>
            <!-- 罗列所有的映射文件 -->

            <mapping resource="News.hbm.xml"/>
        </session-factory>
      </hibernate-configuration>
```

19.1.3　书写第一个 Hibernate 程序

在接下来的内容中，将编写一个 Hibernate 程序，让读者朋友体会 Hibernate 的真正意义。本程序代码保存在（光盘：源代码/第 19 章/19-1/）中，下面开始讲解这个程序的实现流程。

1）虽然在 Hibernate 里有许多 Jar 文件，但是一般不用全部加载就能实现需要的功能。所以首先需要加载 Hibernate 需要的 Jar 文件，如图 19-13 所示。

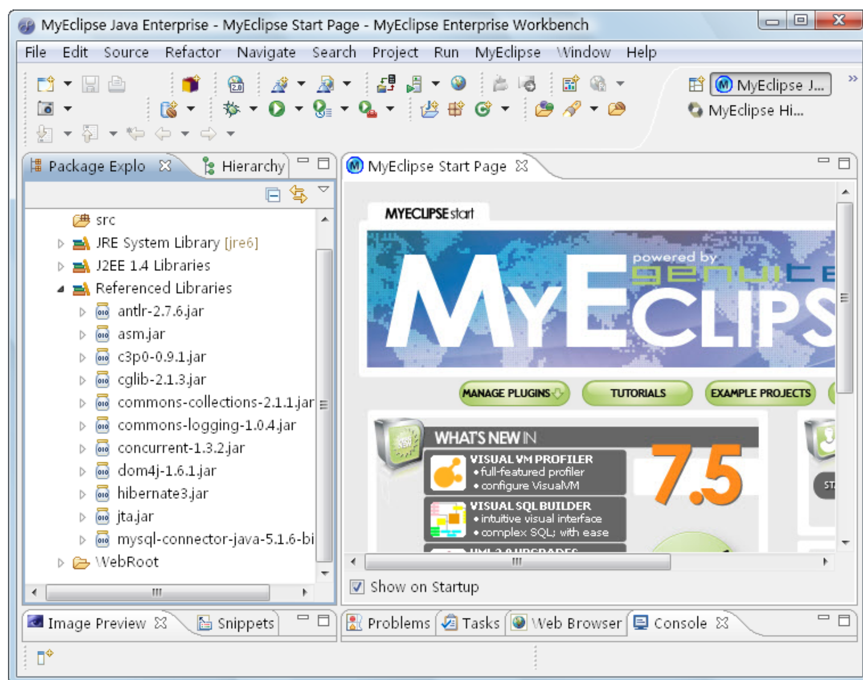


图 19-13　加载需要的 Jar 文件

2）加载文件后，需要建立一个数据库表。在本书光盘中提供了 SQL 语句，读者只需要将 SQL 复制进去并执行后即可得到数据库。然后新建一个名为 bulid.xml 的文件，其页面代码如下。

```
<?xml version="1.0" encoding="GBK"?>
<project name="hibernte" basedir="." default="">
   <property name="src" value="src"/>
   <property name="dest" value="classes"/>
    <path id="classpath">
       <fileset dir="..\..\lib">
          <include name="*.jar"/>
       </fileset>
       <pathelement path="${dest}"/>
    </path>
   <target name="compile" description="Compile all source code">
        <delete dir="${dest}"/>
        <mkdir dir="${dest}"/>
        <copy todir="${dest}">
          <fileset dir="${src}">
                   <exclude name="**/*.java"/>
              </fileset>
        </copy>
       <javac destdir="${dest}" debug="true"
          deprecation="false" optimize="false" failonerror="true">
```

```
                <src path="${src}"/>
                <classpath refid="classpath"/>
            </javac>
        </target>
        <target  name="run"  description="Run  the  main  class"  depends=
"compile">
            <java  classname="lee.ProductManager"  fork="yes"  failonerror=
"true">
                <classpath refid="classpath"/>
            </java>
        </target>
    </project>
```

3）新建文件 hibernate.cfg.xml，其代码如下。

```
    <?xml version="1.0" encoding="GBK"?>
    <!-- 指定 Hibernate 配置文件的 DTD 信息 -->
    <!DOCTYPE hibernate-configuration PUBLIC
      "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
      "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
    <!-- hibernate- configuration 是连接配置文件的根元素 -->
    <hibernate-configuration>
      <session-factory>
            <!-- 指定连接数据库所用的驱动 -->
            <property name="connection.driver_class">com.mysql.jdbc.Driver
</property>
            <!-- 指定连接数据库的 url，hibernate 连接的数据库名 -->
            <property  name="connection.url">jdbc:mysql://localhost/hibernate
</property>
            <!-- 指定连接数据库的用户名 -->
            <property name="connection.username">root</property>
            <!-- 指定连接数据库的密码 -->
            <property name="connection.password">1234</property>
            <!-- 指定连接池里最大连接数 -->
            <property name="hibernate.c3p0.max_size">20</property>
            <!-- 指定连接池里最小连接数 -->
            <property name="hibernate.c3p0.min_size">1</property>
            <!-- 指定连接池里连接的超时间-->
            <property name="hibernate.c3p0.timeout">5000</property>
            <!-- 指定连接池里最大缓存多少个 Statement 对象 -->
            <property name="hibernate.c3p0.max_statements">100</property>
            <property name="hibernate.c3p0.idle_test_period">3000 </property>
            <property name="hibernate.c3p0.acquire_increment">2</property>
            <property name="hibernate.c3p0.validate">true</property>
            <!-- 指定数据库方言 -->
            <property  name="dialect">org.hibernate.dialect.MySQLInnoDBDialect
</property>
```

```
        <!-- 根据需要自动创建数据库 -->
        <property name="hbm2ddl.auto">update</property>
        <!-- 显示 Hibernate 持久化操作所生成的 SQL -->
       <property name="show_sql">true</property>
        <!-- 将 SQL 脚本进行格式化后再输出 -->
       <property name="hibernate.format_sql">true</property>
        <!-- 罗列所有的映射文件 -->
        <mapping resource="Category.hbm.xml"/>
        <mapping resource="Product.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```

4）新建文件 Category.hbm.xml，其代码如下。

```
<?xml version="1.0" encoding="GBK"?>
<!-- 指定 Hibernate 映射文件的 DTD 信息 -->
<!DOCTYPE hibernate-mapping PUBLIC
   "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
   "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping package="lee">
    <class name="Category" table="category_table">
        <!-- 映射标识属性 -->
        <id name="id" column="category_id">
             <!-- 指定 identity 的主键生成器策略 -->
             <generator class="identity"/>
        </id>
        <!-- property 映射普通属性 -->
        <property name="name" type="string"/>
      <property name="effectiveStartDate"
             column="eff_start_date" type="java.util.Date"/>
      <property name="effectiveEndDate"
             column="eff_end_date" type="java.util.Date"/>
        <!-- 映射 Category 和 Product 的关联关系 -->
      <set inverse="true" name="products" table="product_category">
           <key column="category_id"/>
           <many-to-many column="product_id" class="Product"/>
      </set>
        <!-- 对 Category 应用数据过滤 -->
      <filter name="effectiveDate"
             condition=":asOfDate BETWEEN eff_start_date and eff_end_
date"/>
    </class>
</hibernate-mapping>
```

5）新建文件 product.hbm.xml，其代码如下。

```
<?xml version="1.0" encoding="GBK"?>
```

```
        <!-- 指定 Hibernate 映射文件的 DTD 信息 -->
        <!DOCTYPE hibernate-mapping PUBLIC
          "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
          "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
        <hibernate-mapping package="lee">
          <!-- 映射 Course 实体 -->
          <class name="Product" table="product_table">
               <!-- 映射标识属性 -->
               <id name="id" column="product_id">
                    <!-- 指定 identity 的主键生成器策略 -->
                    <generator class="identity"/>
               </id>
               <!-- property 映射普通属性 -->
               <property name="name" column="product_name"
                    type="string"/>
               <property name="stockNumber" column="stock_number"
                    type="int"/>
               <property name="effectiveStartDate"
                    column="eff_start_date" type="java.util.Date"/>
               <property name="effectiveEndDate"
                    column="eff_end_date" type="java.util.Date"/>
               <!-- 映射 Product 和 Category 的关联关系 -->
               <set name="categories" fetch="join" table="product_category" >
                    <key column="product_id"/>
                    <many-to-many column="category_id"
                         class="Category" fetch="join">
                         <!-- 对该关联实体的抓取使用数据过滤 -->
                         <filter name="effectiveDate"
                              condition=":asOfDate BETWEEN
                              eff_start_date and eff_end_date"/>
                         <filter name="category"
                              condition="category_id = :catId"/>
                    </many-to-many>
               </set>
               <!-- 对 Product 实体使用数据过滤 -->
               <filter name="effectiveDate"
                    condition=":asOfDate    BETWEEN    eff_start_date    AND
eff_end_date"/>
          </class>
          <!-- 下面定义 2 个 Filter -->
          <filter-def name="effectiveDate">
               <filter-param name="asOfDate" type="date"/>
          </filter-def>
          <filter-def name="category">
               <filter-param name="catId" type="long"/>
          </filter-def>
```

```
</hibernate-mapping>
```

6）新建 Category 类，具体代码（Category.java）如下。

```
package lee;
import java.util.*;
public class Category
{
    private Long id;
    private String name;
    private Date effectiveStartDate;
    private Date effectiveEndDate;
    private Set<Product> products =
         new HashSet<Product>();
    //无参数的构造器
    public Category()
    {
    }
    //初始化全部属性的构造器
    public Category(String name ,
         Date effectiveStartDate , Date effectiveEndDate)
    {
         this.name = name;
         this.effectiveStartDate = effectiveStartDate;
         this.effectiveEndDate = effectiveEndDate;
    }
    //id 属性的 setter 和 getter 方法
    public void setId(Long id)
    {
         this.id = id;
    }
    public Long getId()
    {
         return this.id;
    }
    //name 属性的 setter 和 getter 方法
    public void setName(String name)
    {
         this.name = name;
    }
    public String getName()
    {
         return this.name;
    }

    //effectiveStartDate 属性的 setter 和 getter 方法
    public void setEffectiveStartDate(Date effectiveStartDate)
```

```
        {
            this.effectiveStartDate = effectiveStartDate;
        }
        public Date getEffectiveStartDate()
        {
            return this.effectiveStartDate;
        }

        //effectiveEndDate 属性的 setter 和 getter 方法
        public void setEffectiveEndDate(Date effectiveEndDate)
        {
            this.effectiveEndDate = effectiveEndDate;
        }
        public Date getEffectiveEndDate()
        {
            return this.effectiveEndDate;
        }
        //products 属性的 setter 和 getter 方法
        public void setProducts(Set<Product> products)
        {
            this.products = products;
        }
        public Set<Product> getProducts()
        {
            return this.products;
        }
        //判断两个物品是否相等
        public boolean equals(Object c)
        {
            if ( this == c )
            {
                return true;
            }
            if(c != null && c.getClass() == Category.class && c.hashCode()
== hashCode())
            {
                return true;
            }
            return false;
        }

        //生成 hashCode 的方式以名字的 hashCode 加上生效开始日期 hashCode 再加生效结束
        //日期的 hashCode
        public int hashCode()
        {
            int result;
```

```
            result = name.hashCode();
            result = 29 * result + (effectiveStartDate != null ?
                 effectiveStartDate.hashCode() : 0);
            result = 29 * result + (effectiveEndDate != null ?
                 effectiveEndDate.hashCode() : 0);
            return result;
        }
    }
```

7）新建一个 HibernateUtil 类，其代码（HibernateUtil.java）如下。

```
package lee;
import org.hibernate.*;
import org.hibernate.cfg.*;
public class HibernateUtil
{
    //使用一个 final 变量来保存不可变的 SessionFactory
    public static final SessionFactory sessionFactory;
    static
    {
         try
         {
              //采用默认的 hibernate.cfg.xml 来启动一个 Configuration 的实例
              Configuration conf = new Configuration()
                   .configure();
              //由 conf 实例来创建一个 SessionFactory 实例
              sessionFactory = conf.buildSessionFactory();
         }
         catch (Throwable ex)
         {
              //
              System.err.println("初始化 SessionFactory 出现异常：" + ex);
              throw new ExceptionInInitializerError(ex);
         }
    }
    //ThreadLocal 是隔离多个线程的数据共享，
    //不存在多个线程之间共享资源,因此不再需要对线程同步
    public static final ThreadLocal session = new ThreadLocal();
    public static Session currentSession()
         throws HibernateException
    {
         Session s = (Session) session.get();
         //如果该线程还没有 Session,则创建一个新的 Session
         if (s == null)
         {
              s = sessionFactory.openSession();
              //将获得的 Session 变量存储在 ThreadLocal 变量 session 里
```

```
                session.set(s);
            }
            return s;
        }
        public static void closeSession()
            throws HibernateException
        {
            Session s = (Session) session.get();
            if (s != null)
            s.close();
            session.set(null);
        }
    }
```

8）新建一个 Product 类，其代码（Product.java）如下。

```
    package lee;
    import java.util.*;
    public class Product
    {
        //定义标识属性
        private Long id;
        //定义产品名
            private String name;
        //定义股票号属性，该属性可标识该产品
            private int stockNumber;
        //定义生效的时间
            private Date effectiveStartDate;
        //定义失效时间
        private Date effectiveEndDate;
        //定义该产品所属的种类
        private Set<Category> categories =
            new HashSet<Category>();
        //无参数的构造器
        public Product()
        {
        }
        //初始化全部属性的构造器
        public Product(String name , int stockNumber ,
            Date effectiveStartDate, Date effectiveEndDate)
        {
            this.name = name;
            this.stockNumber = stockNumber;
            this.effectiveStartDate = effectiveStartDate;
            this.effectiveEndDate = effectiveEndDate;
        }
        //id 属性的 setter 和 getter 方法
```

```
        public void setId(Long id)
        {
            this.id = id;
        }
        public Long getId()
        {
            return this.id;
        }
        //name 属性的 setter 和 getter 方法
        public void setName(String name)
        {
            this.name = name;
        }
        public String getName()
        {
            return this.name;
        }
        //stockNumber 属性的 setter 和 getter 方法
        public void setStockNumber(int stockNumber)
        {
            this.stockNumber = stockNumber;
        }
        public int getStockNumber()
        {
            return this.stockNumber;
        }
        //effectiveStartDate 属性的 setter 和 getter 方法
        public void setEffectiveStartDate(Date effectiveStartDate)
        {
            this.effectiveStartDate = effectiveStartDate;
        }
        public Date getEffectiveStartDate()
        {
            return this.effectiveStartDate;
        }
        //effectiveEndDate 属性的 setter 和 getter 方法
        public void setEffectiveEndDate(Date effectiveEndDate)
        {
            this.effectiveEndDate = effectiveEndDate;
        }
        public Date getEffectiveEndDate()
        {
            return this.effectiveEndDate;
        }
        //categories 属性的 setter 和 getter 方法
        public void setCategories(Set<Category> categories)
```

```
        {
            this.categories = categories;
        }
        public Set<Category> getCategories()
        {
            return this.categories;
        }

        public boolean equals(Object obj)
        {
            if (this == obj)
            {
                return true;
            }
            if (obj != null
                && obj.getClass() == Product.class)
            {
                Product target = (Product)obj;
                if (target.getStockNumber() == stockNumber)
                {
                    return true;
                }
            }
            return false;
        }
        //根据 stockNumber 来生成 hashCode 方法
        public int hashCode()
        {
            return stockNumber;
        }

        //为产品添加一个所属类型
        public void addCategory(Category category)
        {
            if ( category == null )
            {
                return;
            }
            categories.add(category);
            category.getProducts().add(this);
        }
    }
```

9）新建一个 ProductManager 类，其代码（ProductManager.java）如下。

```
    package lee;
    import org.hibernate.Transaction;
```

```
import org.hibernate.Session;
import java.util.*;
import java.text.SimpleDateFormat;
public class ProductManager
{
  SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
  public static void main(String[] args) throws Exception
  {
       ProductManager mgr = new ProductManager();
       mgr.test();
       HibernateUtil.sessionFactory.close();
  }
  private void test() throws Exception
  {
       Session session = HibernateUtil.currentSession();
       Transaction tx = session.beginTransaction();
       //启动 effectiveDate 过滤器，并设置参数
       session.enableFilter("effectiveDate")
            .setParameter("asOfDate", new Date());
       //启动 category 过滤器，并设置参数
       session.enableFilter("category")
            .setParameter("catId", new Long(2));
       //查询所有 Product 实体，不加任何筛选条件
       Iterator results = session.createQuery("from Product as p")
            .iterate();
       while (results.hasNext())
       {
            Product p = (Product)results.next();
            System.out.println(p.getName());
            Iterator it = p.getCategories().iterator();
            System.out.println(p.getCategories().size());
            while (it.hasNext())
            {
                 Category c = (Category)it.next();
                 System.out.println(c.getName());
            }
       }
       tx.commit();
       HibernateUtil.closeSession();
  }
}
```

在上面的代码中，通过 Hibernate 实现了数据过滤功能。数据过滤功能并不是简单的查询操作，而是一种对数据进行筛选的过程。在上面的代码中，定义并使用了过滤器，在代码中启用了过滤器。在大多数情况下，Hibernate 用于写 xml 文档。一旦完成后，Java 就可以将其作为对象使用。在上述文件 product.hbm.xml 的代码中，定义了两个过滤器。而在文件

ProductManager.java 中启用了过滤器。上述代码没有执行结果，因为只定义了一些功能。要它显示信息，必需与 Struts 2 结合。因为 Struts 2 是表现层，而 Hibernate 的功能也只会在持久层操作。

提示：Hibernate 的功能很多，读者可以从下载到的文档中寻找到中文手册，进行阅读，从而对 Hibernate 有一个系统的理解。在上述实例中，只是简单介绍和部署了一个 Hibernate 环境。要想真正深入熟悉 Hibernate 应用，读者需要仔细阅读使用手册。

19.2 数据之间的类型的对应关系

要想深入理解 Hibernate 的映射关系，一定要明白 Java 的数据类型、数据库的数据类型和 Hibernate 数据类型之间的关系。它们之间的关系如表 19-1 所示。

表 19-1 数据类型之间的对应关系

Hibernate 映射类型	Java 类型	标准 SQL 类型
integer	java.lang.Integer	INTEGER
long	java.lang.Long	BIGINT
short	java.lang.Short	SMALLINT
float	java.lang.Float	FLOAT
double	java.lang.Double	DOUBLE
big_decimal	java.math.BigDecimal	NUMERIC
character	java.lang.String	CHAR(1)
string	java.lang.String	VARCHAR
byte	byte 或 java.lang.Byte	TINYINT
boolean	boolean 或 java.lang.Boolean	BIT
yes_no	boolean 或 java.lang.Boolean	CHAR(1)('Y'或'N')
true_false	boolean 或 java.lang.Boolean	CHAR(1)('Y'或'N')
date	java.util.Date 或 java.sql.Date	DATE
time	java.util.Date 或 java.sql.Time	TIME
timestamp	java.util.Date 或 java.sql.Timestamp	TIMESTAMP
calendar	java.util.Calendar	TIMESTAMP
calendar_date	java.util.Calendar	DATE
binary	byte[]	VARBINARY 或 BLOB
text	java.lang.String	CLOB
serializable	java.io.Serializable 实例	VARBINARY 或 BLOB
clob	java.sql.Clob	CLOB
blob	java.sql.Blob	BLOB
class	java.lang.Class	VARCHAR
locale	java.util.Locale	VARCHAR
timezone	java.util.TimeZone	VARCHAR
currency	java.util.Currency	VARCHAR

下面在 MySQL 数据库中建立一张表，其字段和数据类型如表 19-2 所示。

表 19-2　字段和数据类型

字 段 名 称	数 据 类 型	主　键	允 许 为 空	描　述
ID	int(4)	是		ID 号
MYBOOLEAN	bit(1)		√	逻辑型数据
MYINT	int(5)		√	整型数据
MYLONG	bigint(11)		√	长整型数据
MYFLOAT	float(8,2)		√	单精度浮点型数据
MYDOUBLE	double(10,2)		√	双精度浮点型数据
MYDECIMAL	decimal(10,2)		√	DECIMAL 型数据
MYSTRING	varchar(100)		√	字符串数据
MYTEXT	text		√	Text 型数据
MYDATE	date		√	Date 型数据
MYTIME	time		√	Time 型数据
MYDATETIME	datetime		√	Datetime 型数据
MYTIMESTAMP	mestamp		√	Timestamp 型数据
MYBINARY	varbinary(10240)		√	Binary 型数据
MYBLOB	longblob		√	Blob 型数据

在文件 Datamap.java 中新建一个 Datamap 类，其源代码（光盘：源代码/第 19 章/19-2/）如下。

```
package com.ORM;
import java.io.Serializable;
public class Datamap implements Serializable{
private int hashValue = 0;
private java.lang.Integer id;
private java.lang.Boolean myboolean;
private java.lang.Integer myint;
private java.lang.Long mylong;
private java.lang.Float myfloat;
private java.lang.Double mydouble;
private java.math.BigDecimal mydecimal;
private java.lang.String mystring;
private java.lang.String mytext;
private java.sql.Date mydate;
private java.sql.Time mytime;
private java.util.Date mydatetime;
private java.sql.Timestamp mytimestamp;
private byte[] mybinary;
private java.sql.Blob myblob;
public Datamap() {
}
```

新建配置文件 hibernate.cfg.xml，其实现代码如下。

```
<?xml version="1.0" encoding='UTF-8'?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN""http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd" >
<hibernate-mapping package="com.ORM">
<class name="Datamap" table="datamap">
<id name="id" column="ID" type="integer">
<generator class="identity"/></id>
<property name="myboolean" column="MYBOOLEAN" type="boolean" />
<property name="myint" column="MYINT" type="integer" />
<property name="mylong" column="MYLONG" type="long" />
<property name="myfloat" column="MYFLOAT" type="float" />
<property name="mydouble" column="MYDOUBLE" type="double" />
<property name="mydecimal" column="MYDECIMAL" type="big_decimal" />
<property name="mystring" column="MYSTRING" type="string" />
<property name="mytext" column="MYTEXT" type="text" />
<property name="mydate" column="MYDATE" type="date" />
<property name="mytime" column="MYTIME" type="time" />
<property name="mydatetime" column="MYDATETIME" type="timestamp" />
<property name="mytimestamp" column="MYTIMESTAMP" type="timestamp" />
<property name="mybinary" column="MYBINARY" type="binary" />
<property name="myblob" column="MYBLOB" type="blob" />
</class>
</hibernate-mapping>
```

在文件 TestDA.java 中新建 TestDAO 类，其实现代码如下。

```
package com.ORM;
import com.ORM.*;
 public interface TestDAO
{
public void addDatamap(Datamap datamap);
public Datamap loadDatamap(Integer id);
public void delDatamap(Integer id);
}
```

在文件 TestDAOImpl.java 中新建 TestDAOImpl 类，其实现代码如下。

```
package com.DAO;
import com.ORM.*;
import org.hibernate.*;
public class TestDAOImpl implements TestDAO {
public void addDatamap(Datamap datamap) {
Session session = MySessionFactory.currentSession();
Transaction ts = null;
 try{
```

```
 ts = session.beginTransaction();
session.save(datamap);
ts.commit();
}catch(Exception ex){
if(ts!=null)ts.rollback();
System.out.println("【系统错误】 在 TestDAOImpl 的 addDatamap 方法中出错：");
ex.printStackTrace();
}finally{
   MySessionFactory.closeSession();
   }
}
public Datamap loadDatamap(Integer id) {
Session session = MySessionFactory.currentSession();
Transaction ts = null;  Datamap datamap = null;
try{  ts = session.beginTransaction();
datamap = (Datamap)session.get(Datamap.class,id);  ts.commit();  }
catch(Exception ex){  if(ts!=null)ts.rollback();
System.out.println("【系统错误】  在 TestDAOImpl 的 loadDatamap 方法中出错：");
ex.printStackTrace();  }finally{  MySessionFactory.closeSession();  }
return datamap;  }
 public void delDatamap(Integer id) {
Session session = MySessionFactory.currentSession();
Transaction ts = null;
try{  ts = session.beginTransaction();
 Datamap datamap = (Datamap)session.load(Datamap.class,id);
 session.delete(datamap);
ts.commit();
}catch(Exception ex){
if(ts!=null)ts.rollback();
System.out.println("【系统错误】 在 TestDAOImpl 的 delDatamap 方法中出错：");
ex.printStackTrace();  }finally{  MySessionFactory.closeSession();
}
}
 }
```

新建文件 TestBean.java，其代码如下。

```
import com.DAO.*;
import com.ORM.*;
import java.io.*;
import java.math.BigDecimal;
import java.net.*;
import org.hibernate.*;
public class TestBean {
TestDAO dao = new TestDAOImpl();  //得到指定 URL 的 HTML 内容
private String getHtmlByUrl(String url){
StringBuffer html = new StringBuffer();
```

```
        String line = null;  try{  URL u = new URL(url);
        URLConnection uc = u.openConnection();
        BufferedReader  br  =  new  BufferedReader    (new  InputStreamReader
(uc.getInputStream()));
        while ((line=br.readLine())!=null){  html.append(line);
        }
        }catch(Exception ex){
        System.out.println("【系统错误】 在 TestBean 的 getHtmlByUrl 方法中出错：");
        ex.printStackTrace();  }
        return html.toString();  }
        //读取二进制流
        private byte[] readBinary(InputStream in){
        byte[] binCodes = null;
        try{  binCodes = new byte[in.available()];
         in.read(binCodes);  in.close();
        }catch(Exception ex){  System.out.println("【系统错误】  在 TestBean 的
readBinary 方法中出错：");  ex.printStackTrace();
        }  return binCodes;
        }
        //从输入流创建 BLOB 对象
        private java.sql.Blob getBlob(InputStream in){
        java.sql.Blob blob = null;
        try{  blob = Hibernate.createBlob(in);
        in.close();
        }catch(Exception ex){
        System.out.println("【系统错误】 在 TestBean 的 getBlob 方法中出错：");
        ex.printStackTrace();
        }  return blob;  }
        public void addDatamap(){
        Datamap datamap = new Datamap();
        datamap.setMyboolean(new Boolean(true));
        datamap.setMyint(new Integer(100));
        datamap.setMylong(new Long(12345678));
        datamap.setMyfloat(new Float(3.14));
        datamap.setMydouble(new Double(0.8628));
        datamap.setMydecimal(new BigDecimal(5200.80));
        datamap.setMystring("这是测试内容");
        datamap.setMytext(getHtmlByUrl("http://www.163.com"));
        datamap.setMydate(new java.sql.Date(System.currentTimeMillis()));
        datamap.setMytime(new java.sql.Time(8,27,56));
        datamap.setMydatetime(new java.util.Date());
        datamap.setMytimestamp(newjava.sql.Timestamp
        (System.currentTimeMillis()));
        datamap.setMybinary(readBinary(getClass().
getResourceAsStream("TestBean.class")));
        datamap.setMyblob(getBlob(getClass().
```

```
getResourceAsStream("/photo1.jpg")));
        dao.addDatamap(datamap); }
        public Datamap loadDatamap(Integer i){
        return dao.loadDatamap(i);
        }
        public void delDatamap(Integer i){
           dao.delDatamap(i);
           }
        }
```

新建文件 index.jsp，具体代码如下。

```
        <%@ page language="java" pageEncoding="gb2312"%>
        <%@ page import="com.bean.TestBean"%>
        <%@ page import="com.ORM.*"%>
        <%@ page import="java.io.*"%>
        <jsp:useBean id="test" class="com.bean.TestBean" />
        <html>
         <head>
         <title>Hibernate 映射数据类型</title>
         </head>
        <body>
        <h2>Hibernate 映射数据类型</h2><hr>
         <%  test.addDatamap();
         Integer cur = new Integer(1);
        Datamap datamap = test.loadDatamap(cur);
        out.println("datamap.getMyboolean()="+datamap.getMyboolean());
        out.println("<br>datamap.getMyint()="+datamap.getMyint());
        out.println("<br>datamap.getMylong()="+datamap.getMylong());
        out.println("<br>datamap.getMyfloat()="+datamap.getMyfloat());
        out.println("<br>datamap.getMydouble()="+datamap.getMydouble());
        out.println("<br>datamap.getMydecimal()="+datamap.getMydecimal());
        out.println("<br>datamap.getMystring()="+datamap.getMystring());
         out.println("<br>datamap.getMydate()="+datamap.getMydate());
        out.println("<br>datamap.getMytime()="+datamap.getMytime());
        out.println("<br>datamap.getMydatetime()="+datamap.getMydatetime());
        out.println("<br>datamap.getMytimestamp()="+datamap.getMytimestamp());  %>
          <br>datamap.getMytext():<br>
          <textarea cols="80" name="textfield" rows="10">
           <%=datamap.getMytext()%>
            </textarea>
            <br>datamap.getMybinary():<br>
        <textarea cols="80" name="textfield" rows="10">
         <%   byte[]  binCodes  =  datamap.getMybinary();out.print(new  String
(binCodes));  %>
        </textarea>  <br>datamap.getMyblob():<br>
        <textarea cols="80" name="textfield" rows="10">
```

```
<%  InputStream in = datamap.getMyblob().getBinaryStream();
int tmp = 0;
while((tmp=in.read())!=-1){  out.print(tmp);  }  %>
</textarea>
<%test.delDatamap(cur);%>
</body>
  </html>
```

运行上述代码，得到如图 19-14 所示的效果。

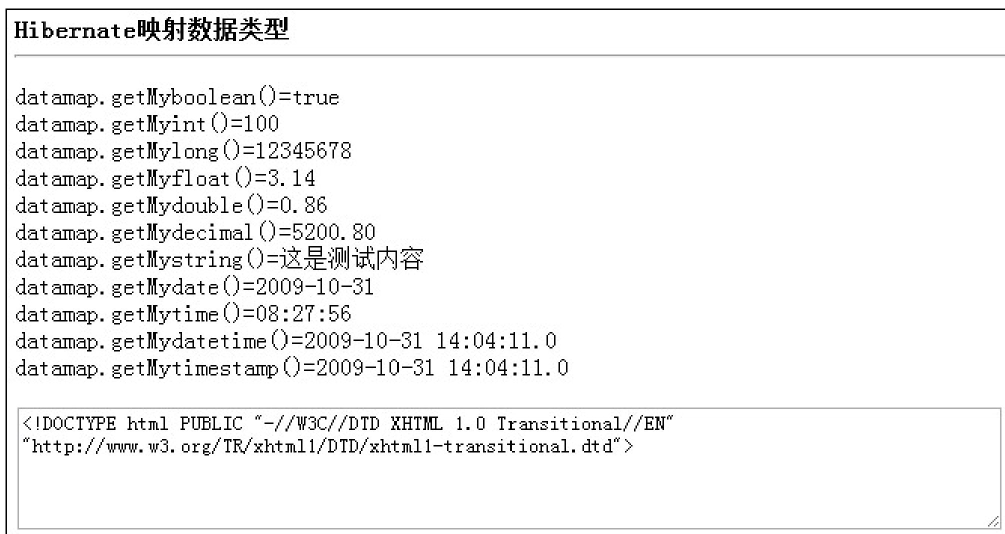


图 19-14　运行的效果

19.3　Hibernate 的基本映射

Hibernate 的基本映射数据类型很重要，是 Java 基本类型与标准 SQL 类型相互转换的桥梁。通过 Hibernate 的基本映射数据类型，可以方便地将数据从一种形式转换成另一种形式，完成高质量的 ORM 任务。

19.3.1　主键生成器

主键生成器的功能是生成数据表记录的主键，通常有如下 4 种常见的主键生成器。

1）Increment：对 long，short 或 int 的数据列生成自动增长主键。

2）Identity：对如 SQL Server，MySQL 等支持自动增长列的数据库，如果数据列的类型是 long，short 或 int，可使用主键生成器生成自动增长主键。

3）Sequence：对如 Oracle，DB2 等支持 Sequence 的数据库，如果数据列的类型是 long，short 或 int，可使用该主键生成器生成自动增长主键。

4）Uuid：对字符串列的数据采用 128 位 uuid 算法生成唯一的字符串主键。

19.3.2　映射集合属性

因为映射是一个有序集合，所以在持久化到数据库时必须增加一列以表示集合元素的次序。

1. List 集合属性

例如，在下面的持久化类中，该 Person 类有一个集合属性 schools，该属性对应多个学校。因为集合属性只能以接口声明，所以在下面的代码中，schools 的类型只能是 List，而不能是 ArrayList，该集合属性必须使用实现类完成初始化。

```
public class Person implements Serializable
{
     II 标识属性
     private int id;
     II 名字属性
     private String name;
     II 集合属性，学校
     private List schools = new ArrayList();
}
```

在该持久化类中，其普通属性的映射与前面的基本相同，唯一的差别是增加了集合属性。在本例中的 List 集合属性，应该使用 List 元素实现映射。List 元素规定用 list-index 的子元素来映射有序集合的次序列。因为集合属性的值会存放在另外的表中，不可能与持久化类存储在同一个表内，所以必须以外键关联，用 key 元素来映射该外键列。

下面是该持久化类的映射文件的实现代码。

```
<?xml vers工on= 1I 1.0"?>
<!--Hibernate 映射文件的文件头，包含 DTD 等信息-->
<!DOCTYPE hibernate-mapping
PUBL工C "-IIHibernatel 丑工bernate Mapping DTD 3.01IEN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<! -- H 工bernate 映射文件的根元素-->
<hibernate-mapping package="lee">
<!-- 每个 class 元素映射一个持久化类>
<class name="Person">
<!-- 定义持久化类的标识属性-->
<id n缸ne="id" column="personid">
<!-- 定义主键生成器策略-->
<generator class="identity"l>
<lid>
<!--映射普通属性 name-->
<property name="name"l>
<!--映射普通属性 age-->
<property name="age"l>
<!- 映射集合属性 schools，指定存放集合属性的表名->
```

```
<list name="schools" table="school">
<!- 集合属性和持久化类的关联外键>
<key column="personid" not-null="true"l>
<!-- 集合属性的次序列-->
<list-index column="list_order"l>
<!-- 集合属性的元素-->
<element type="string" column="school_name"l>
</list>
</class>
</hibernate-mapping>
```

2. Set 集合属性

Set 集合属性的映射与 List 非常相似，因为 Set 是无序的、不可重复的集合，所以 Set 元素无需使用 index 元素来指定集合元素的次序。与 List 一样，Set 集合也需要外键列，用于持久化类和集合属性的关联。读者可将上面代码中的 List 集合属性改为 Set 集合属性，看一看具体效果，在本书将不再赘述。

映射文件与 List 集合属性的映射相似，差别是当使用 Set 元素时，无需增加 index 列来保存集合元素的次序。下面是 Set 集合属性的映射文件的实现代码。

```
<?xml version="1.0"?>
<!-- Hibernate 的映射文件的文件头，包含 DTD 等信息--〉
<!DOCTYPE hibernate-mapping
PUBLIC "< IIHibernate/Hibernate Mapp工ng DTD 3.01IEN"
''http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!-- Hibernate 映射文件的根元素-->
<hibernate-mapp工ng package="lee">
<!--每个 class 元素映射→个持久化类-->
<class name="Person">
<!-- 映射标识属性-->
<id name="id" column="personid">
<'--指定主键生成器-->
<generator class="identity"l>
<lid>
<!--映射 name 属性-->
<property name="name"l>
<!--映射 age 属性>
<property name="age"l>
<!-- 映射 Set 集合属性，指定集合属性存入 school 表…>
<set name="schools" table="school">
<1 二集合属性的外键列-->
<key column="personid" not-null="true"l>
<!--用于映射集合中的元素>
<element type="string" column="school_name" not-null="true"l>
</set>
</class>
</hibernate-mapping>
```

3．bag 元素映射

bag 元素很重要，它不但可以为 List 集合属性映射，也可以为 Collection 集合属性映射。不管是哪种集合属性，使用 bag 元素后都将被映射成无序集合，而集合属性对应的表没有主键。bag 元素只需要使用 key 元素来映射关联的外键列，使用 element 元素来映射集合属性的每个元素。下面是持久化类使用 bag 元素的映射文件的实现代码。

```
<?xml version="1.0"?>
< J -- Hibernate 映射文件的文件头，包含 DTD 等信息--
<!DOCTYPE hibernate-mapping
PUBLIC "-IIHibernate/Hibernate Mapping DTD 3.01IEN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<! -- Hibernate 映射文件的根元素-->
<hibernate-mapping package="lee">
<!--每个 class 元素映射一个持久化类-->
<class name="Person">
<! -- id 元素映射标识属性-->
<id name="id" column="personid">
<'--指定主键生成器策略…>
<generator class="identity"l>
<lid>
<!-- 映射 name 属性-->
<property name="name"/>
<!-- 映射 age 属性…>
<property name="age"l>
<!-- 映射集合属性-->
<bag name="schools" table="school">
<key column="personid" not-null="true"/>
<element type="string" column="school_name"/>
</bag>
</class>
</hibernate-mapping>
```

4．Map 集合属性

Map 集合属性不仅需要映射属性 value，还需要映射属性 key。当映射 Map 集合属性时，也需要指定外键列，另外还必须指定 Map 的 key 列，并且系统将以外键列和 key 列作为联合主键。与所有集合属性类似的是，只能使用接口声明集合属性，看下面的 POJO 类。

```
<?xml version="1.0"?>
<!-- Hibernate 映射文件的文件头，包含 DTD 信息-->
<!DOCTYPE hibernate-mapping  PUBL 工 C "-//Hibernate/Hibernate Mapping
DTD 3.0//EN"
''http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!--Hibernate 映射文件的根元素-->
<hibernate-mapping package="lee">
<!-- 每个 class 元素映射一个持久化类-->
<class name="Person">
```

```
<!-- id 元素用于映射标识属性>
<id name="id" column="personid">
<!--指定主键生成器策略-->
<generator class="identity"/>
<lid>
<!-- 映射 name 属性-->
<property name="name"/>
<!-- 映射 age 属性…>
<property name="age"/>
<!-- 映射 Map 集合属性>
<map name="scores" table="score">
<!--映射外键 J"tl-->
<key column="personid" not-null="true"/>
<!--映射 Map Key-->
<map-key column="xueke" type="string"/>
<!--映射 Map Value-->
<element column="fenshu" type="float"/>
</map>
</class>
</hibernate-mapping>
```

上述程序运行结束后，会保存集合属性 scores 的表，并以 personid 和 xueke 作为联合主键。

19.3.3 映射引用属性

为了映射引用属性，在 Hibernate 中提供了 component 元素。引用属性必须指定该属性的类型，每个 component 元素映射一个引用属性。 componet 元素要求具有 class 属性，该属性用于确定引用属性的类型。

1. 如何设置引用属性

现实应用中的一个自定义类通常还包括其他属性，所以很有必要为 component 元素增加 property 的子元素，用于映射引用属性的子属性。下面是持久化类的映射文件的实现代码。

```
<?xml version="1.0"?>
<!--Hibernate 映射文件的文件头，包含 DTD 信息-->
<lDOCTYPE hibernate-mapping
PUBLIC "-IIHibernate/Hibernate Mapping DTD 3.OIIEN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<! -- Hibernate 映射文件的文件头…>
<hibernate-mapping package="lee">
<!--每个 class 元素映射一个持久化类-->
<class name="Person" table="person">
<!--id 属性映射标识属性-->
<id name:;;: lI id" colurnn="persoD_id">
<!--指定主键生成器策略-->
<generator class="increment"l>
```

```
<lid>
<!--映射普通属性 age-->
<property name="age"l>
<!--映射引用属性 name，引用属性的类型为 Name ?
<component name="name" class="Name" unique="true">
<!-- 映射引用属性的 f 工 rst 属性-->
<property name="first"/>
<!-- 映射引用属性的 last 属性-->
<property name="last"/>
</component>
</class>
</hibernate-mapping>
```

在上述映射文件中，component 还有 unique=“true”属性并不是必需的，而是与具体的业务逻辑相关联的。

2．集合引用属性映射

如果存在集合属性 POJO，则需要使用 Set、List 和 bag 等集合元素来映射集合属性。如果集合里的元素是普通字符串，则只需使用 element 映射集合元素即可。如果集合元素是自定义类，则必须使用 composite-element 子元素来映射集合元素。由于 composite-element 元素映射一个引用类型，因此需要增加 class 元素来确定集合元素的类型，该元素还支持以 property 的子元素来定义引用类型的子属性。下面是 Person 类的持久化映射文件的实现代码。

```
<?xml version="1.0"?>
<!--Hibernate 映射文件的文件头，包含 DTD 等信息-->
<!DOCTYPE hibernate-mapping
PUBLIC "-IIHibernate/Hibernate Mapping DTD 3.01EN"
''http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!--Hibernate 映射文件的根元素-->
<hibernate-mapping package="lee">
<!--每个 class 元素映射一个持久化类-->
<class name="Person">
<!--id 属性映射…>
<id name="id" column="personid">
<!-- 指定主键生成器策略->
<generator class="identity"l>
<lid>
<!-- 映射普通属性口 ame-->
<property name="name"l>
<!-- 映射普通属性 age-->
<property name="age"l>
<!--映射 List 集合属性-->
<list name="schools" table="school">
<!--映射关联外键列-->
<key column="personid" not-null="true"l>
```

```
<List 有序集合，需要索引列-->
<list-index column="list_order"l>
<!-- composite-element 映射集合里的元素， class 属性确定集合里元素的类型-->
<composite-element class="School">
<!-- 每个 property 属性映射集合元素的基本属性-->
<property name="name"/>
<property name="address"/>
</composite-element>
</list>
</class>
</hibernate-mapping>
```

3．引用类型主键的映射

Person 标识属性既不是基本数据类型，也不是 String 字符串，而是用户自定义的 Name 类型。如果持久化类需要使用引用类型作为表示属性，则该类应该满足如下两个条件。

1）实现 java.io.Seria1izable 接口。

2）重写 equals()和 hashCode()方法：根据数据表中联合主键的列来判断这两个方法的返回值。对于 Person 持久化类，Name 是标识属性的类，Name 实例是可以唯一标识的 Person 实例。根据业务需要，能唯一标识 Person 实例的应该是属性 firstName 和 lastName。因此应该根据属性 firstName 和 lastName 来判断 hashCode 和 equals 方法。当引用类型主键的映射时，应使用 composite-id 元素，该元素需要 class 属性来确定主键的引用类型，并使用 key-property 元素来确定引用类型包含的基本属性。

4．复合主键的映射

映射复合主键的持久化类必须满足如下两个条件。

1）实现 java .io.Serializable 接口。

2）重写 equals()和 hashCode()方法，根据数据表中的联合主键来判断这两个方法的返回值。

如需映射 firstName 和 lastName 两个标识属性，也可使用 composite-id 元素。因为标识属性既没有实现类，也不是一个真实存在的属性，所以 composite-id 元素不需要 name 和 class 属性。下面通过具体代码进行讲解，其代码如下。

```
<?xml version="1.0"?>
<!--Hibernate 映射文件的文件头，包含 DTD 等信息>
<lDOCTYPE hibernate-mapping
PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
''http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!-- Hibernate 映射文件的根元素-->
<hibernate-mapping package="lee">
<!-- 每个 class 元素映射一个持久化类-->
<class name="Person" table="personcomid">
<!-- composite-id 用于映射复合主键-->
<composite-id>
<!-- key-property 元素映射复合主键的每个元素>
```

```
        <key-property name="firstname"/>
        <key-property name="lastname"/>
        </composite-id>
        <property name="age"/>
        </class>
        </hibernate-mapping>
```

19.4　数据库对象映射

在编程过程中，常常会希望创建和删除触发器、存储过程等数据库对象。在 Hibernate 中，提供了对上述功能的支持，在本节的内容中将一一讲解。

19.4.1　如何将普通表转换成持久数据库表

下面将通过几段代码来讲解数据库的映射，首先建立如图 19-15 所示的数据库映射表。

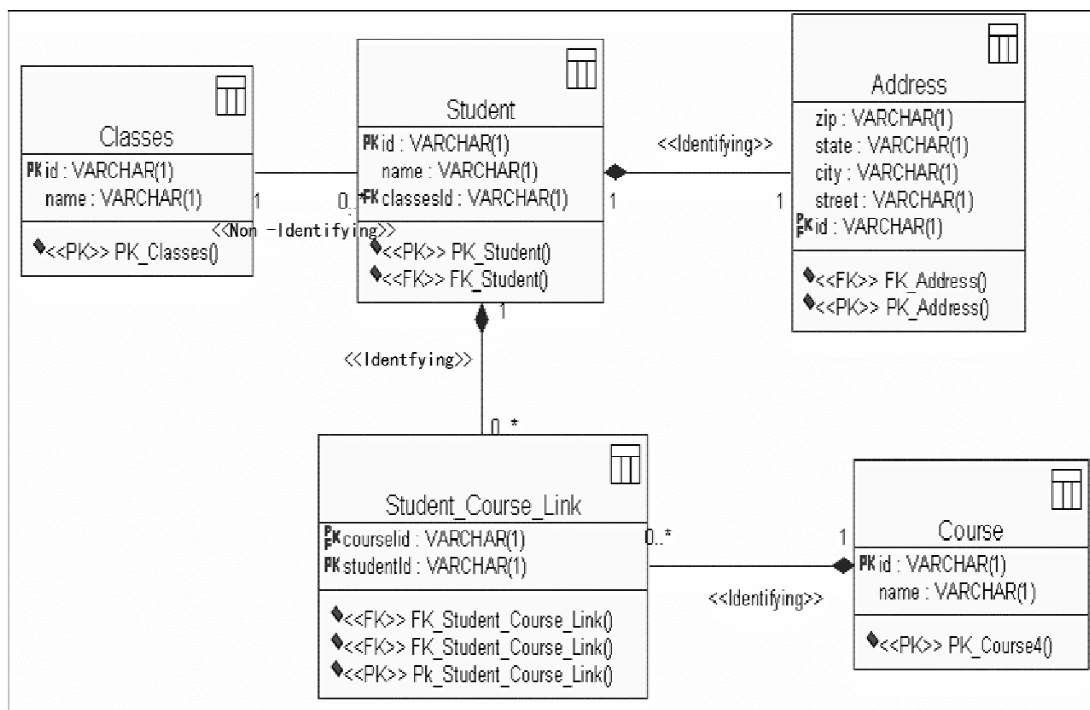


图 19-15　数据库映射表

在图 19-15 所示的数据模型图中，Student 是所有表的核心，它和 Classes 表是一对多的关系，和 Course 表是多对多的关系（通过 Student_Course_Link 表来链接），和 Address 表是一对一的关系。通过数据类型，可以将上面的图转换成持久对象，如图 19-16 所示为持久对象数据库表。

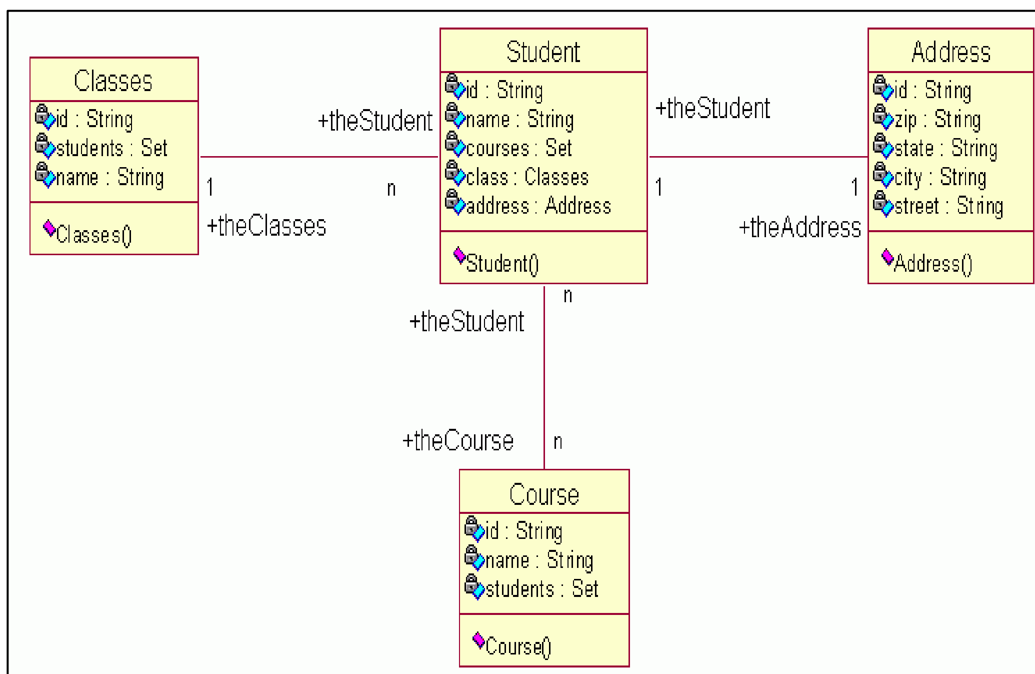


图 19-16　建立持久数据库表

数据模型图和 Java 持久对象的类图虽然很相似，但是不完全相同。比如 Classes 表和 Student 表是一对多的关系，在类图中，两者仍然是一对多的关系；但是在 Classes 类中添加了一个 Student 属性，属性的类型是 java.util.Set，它表示 Classes 对象中包含的所有 Student 对象。

19.4.2　创建 Hibernate 持久对象

接下来以上节建立的数据模型为例来创建持久对象，持久对象之间的关系由图 19-15 所示的类图指定。首先看 Student 类，这是此关系映射的核心。创建一个名为 Student.java 的文件，其代码（光盘：源代码/第 19 章/19-3/）如下。

```
package com.hellking.study.hibernate;
import java.util.Set;
/**
 *在 hibernate 中代表了 Students 表的类。
 */
public class Student
{
   /**属性，和 students 表中的字段对应**/
   private String id;
   private String name;
   /**和其他类之间的映射关系**/
   private Set courses;
   private Classes classes;
```

```
        private Address address;
         /**属性的访问方法,必须是公共的方法**/
         public void setId(String string) {
            id = string;
         }
         public String getId() {
            return id;
         }

         public void setName(String name)
         {
            this.name=name;
         }
         public String getName()
         {
            return this.name;
         }
         /**操作和其他对象之间的关系**/
         public void setCourses(Set co)
         {
            this.courses=co;
         }
         public Set getCourses()
         {
            return this.courses;
         }
         public void setAddress(Address ad)
         {
            this.address=address;
         }
         public Address getAddress()
         {
            return this.address;
         }
         public void setClasses(Classes c)
         {
            this.classes=c;
         }
         public Classes getClasses()
         {
            return this.classes;
         }
    }
```

在 Student 类中，因为 Students 表和 Classes 的表是多对一的关系，所以它包含了一个类型为 Classes 的 classes 属性，表示一个学生可以有一个班级；Students 表和 Address 表是一对

一的关系，同样也包含了一个类型为 Address 的 address 属性，表示一个学生有一个地址；Students 表和 Course 表是多对多的关系，所以包含了一个类型为 java.util.Set 的 course 属性，表示一个学生可以学习多门课程。同样，某个课程可以由多个学生来选修。Classes 对象和 Student 对象是一对多的关系。Classes 对象包含一个类型为 java.util.Set 的 students 属性，具体代码（光盘：源代码/第 19 章/19-3/）如下。

```
package com.hellking.study.hibernate;
import java.util.Set;
/**
 *在 hibernate 中代表了 Classes 表的类。
 */
public class Classes
{
   /**属性，和 classes 表的字段一致**/
   private String id;
   private String name;
   /**和其他类之间的映射关系**/
   private Set students;

   /**属性的访问方法,必须是公共的方法**/
   public void setId(String string) {
       id = string;
    }

    public String getId() {
       return id;
    }
    public void setName(String name)
    {
       this.name=name;
    }
    public String getName()
    {
       return this.name;
    }

    /**操作和其他对象之间的关系**/
    public void setStudents(Set stud)
    {
       this.students=stud;
    }
    public Set getStudents()
    {
       return this.students;
    }
}
```

19.4.3　描述对象之间的关系

编写好了持久对象后，还需要描述它们之间的关系。通过下面的代码讲解如何描述 Student 持久对象的关系，实现文件 Student.hbm.xml 的代码如下。

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
   "-//Hibernate/Hibernate Mapping DTD 2.0//EN"
   "http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd">
<hibernate-mapping>
   <class
      name="com.hellking.study.hibernate.Student"
      table="Students"
      dynamic-update="false"
   >
  <!-- 描述 ID 字段-->
      <id
         name="id"
         column="StudentId"
         type="string"
         unsaved-value="any"
      >
         <generator class="assigned"/>
      </id>
    <!-- 属性-->
       <property
         name="name"
         type="string"
         update="true"
         insert="true"
         column="Name"
      />
<!-- 描述 Student 和 Course 多对多的关系-->
       <set
         name="courses"
         table="Student_Course_Link"
         lazy="false"
         inverse="false"
         cascade="all"
         sort="unsorted"
      >
           <key
              column="StudentId"
           />
           <many-to-many
              class="com.hellking.study.hibernate.Course"
              column="CourseId"
```

```
                    outer-join="auto"
                />
            </set>
        <!-- 描述 Student 和 Classes 之间多对一的关系-->
                <many-to-one
                  name="classes"
                  class="com.hellking.study.hibernate.Classes"
                  cascade="none"
                  outer-join="auto"
                  update="true"
                  insert="true"
                  column="ClassesId"
              />

        <!-- 描述 Student 和 Address 之间一对一的关系-->
         <one-to-one
                  name="address"
                  class="com.hellking.study.hibernate.Address"
                  cascade="none"
                  outer-join="auto"
                  constrained="false"
              />
          </class>
        </hibernate-mapping>
```

在上面的关系描述中描述了如下 3 种关系。

1）第一种关系：Student 和 Address 之间一对一的关系，它是最简单的关系，描述的方法如下。

```
        <one-to-one name="" class="">
```

其中 name 表示 Student 对象中名称为 address 的属性；class 表示 address 属性的类型：com.hellking.study.hibernate.Address。

2）第二种关系：Student 和 Classes 之间多对一的关系，描述的方法如下。

```
        <many-to-one       name="classes"   class="com.hellking.study.hibernate.
Classes" column="ClassesId" …   />
```

其中 name 表示 Student 对象中名称为 classes 的属性；class 表示的是 classes 属性的类型，column 表示 Student 表引用 Classes 表使用的外部键名称。并且在 Classes 类中也引用了 Student 类，它使用了以下的描述符来描述这个关系。

```
         <set
        name="students"
         table="Students"
        lazy="false"
        inverse="false"
```

```
  ascade="all"
sort="unsorted" >
<key  column="ClassesId" />
<one-to-many  class="com.hellking.study.hibernate.Student" />
</set>
```

3）第三种关系：Student 和 Course 之间多对多的关系，使用以下的方法描述。

```
<set
name="coursesld"t
able="Student_Course_Link"
lazy="false"
 inverse="false"
cascade="all"
sort="unsorted" >
<key column="StudentId" />
<many-to-many  class="com.hellking.study.hibernate.Course"
 column="CourseId"
 outer-join="auto" />
</set>
```

在映射多对多关系时，还需要使用一个链接表，table 属性用来指定表名。链接表中包含了两个字段，分别是 CourseId 和 StudentId，描述方法如下。

```
<key column="StudentId">
```

需要指定 Student 对象在 Student_Course_Link 链接表中的外部键，在 Course 持久对象使用了以下的描述符来描述这个关系。

```
<set
name="students"
table="Student_Course_Link"
lazy="false"
inverse="false"
cascade="all"
sort="unsorted" >
<key column="CourseId" />
<many-to-many
class="com.hellking.study.hibernate.Student"
column="StudentId"
outer-join="auto"/>
 </set>
```

由于其他持久对象的描述基本一样，不再赘述。

19.4.4 使用映射关系

持久对象使用最频繁的操作是增加数据、查询数据、删除数据、更新数据。接下来将通

过一个具体实例，来测试映射，实现最基本的操作。

1. 添加数据到数据库

当添加数据到数据库时，要看数据库的添加情况，在此需要添加下面的代码，功能是测试持久对象之间的映射关系，具体代码如下。

```
        /**
        *在数据库中添加数据
        */
        public  void  addData(String  studentId,String  classesId,String
coursesId)
           throws HibernateException {
           try
           {
             /**
              *以下的代码添加了一个 Student，同时为 Student 指定了
              *Address、Courses 和 Classses
              */
             beginTransaction();
             //创建一个 Student 对象
             Student student = new Student();
             student.setName("hellking2");
             student.setId(studentId);

             //创建一个 Address 对象
             Address addr=new Address();
             addr.setCity("beijing");
             addr.setState("bj");
             addr.setStreet("tsinghua");
             addr.setZip("100083");
             addr.setId(student.getId());
               //设置 Student 和 address 的关系
             student.setAddress(addr);

            Set set=new HashSet();
            set.add(student);
            //创建一个 course 对象
            Course course=new  Course  ();
            course.setId(coursesId);
            course.setName("computer_jsp");
            //设置 course 和 student 对象之间的关系
            course.setStudents(set);

            //创建一个 classes 对象
            Classes cl=new Classes();
            cl.setId(classesId);
            cl.setName("engine power");
```

```
            //设置某个 classes 对象包含的 students 对象
            cl.setStudents(set);
           //由于是双向的关系，student 对象也需要设置一次
            student.setClasses(cl);

            //保存创建的对象到 session 中
            session.save(cl);
            session.save(course);
            session.save(student);
            session.save(addr);
            //提交事务，使更改生效
            endTransaction(true);
          }
          catch(HibernateException e)
          {
              System.out.println("在添加数据时出错！");
              e.printStackTrace();
              throw e;
          }
      }
```

添加数据到数据库之前，首先设置持久对象的各个属性，其代码如下。

```
student.setName("hellking2");
```

这种设置属性的方式和普通类没有区别，当设置完所有的属性后，就需要设置持久对象之间的关系，具体代码如下。

```
student.setAddress(addr);
```

如果存在对象之间的多重关系，那么可能需要把对象保存在 Set 集合中，然后再进行设置，具体代码如下。

```
Set set=new HashSet();
set.add(student);
course.setStudents(set);
```

当设置完所有的属性和对象关系之后，就可以调用下面的代码。

```
session.save(persistentObject);
```

把持久对象保存到 Hibernate 会话中。最后，调用 endTransaction 来提交事务，并且关闭 Hibernate 会话。

2．数据查询

在复杂的实体对象映射中，实现查询也比较复杂。在下面的程序里，将为读者提供几种实现数据查询的方法。具体代码如下。

```
    /**
     *获得某个给定 studentid 的 Student 的地址信息
     */
    public Address getAddress(String id) throws HibernateException
    {
           beginTransaction();
           Student st=(Student)session.load(Student.class,id);
           Address addr=(Address)session.load(Address.class,st.getId());
           endTransaction(false);
           return addr;

    }

    /**
     *获得某个给定 studentid 的 Student 的所有课程
     */
    public Set getCourses(String id)throws HibernateException
    {
        beginTransaction();
        Student st=(Student)session.load(Student.class,id);
        endTransaction(false);
        return st.getCourses();
    }

    /**
     *测试获得某个给定 studentid 的 Student 所属的 Classes
     */
    public Classes getClasses(String id)throws HibernateException
    {
        beginTransaction();
        Student st=(Student)session.load(Student.class,id);
        System.out.println(st.getClasses().getId());
        endTransaction(false);
        return st.getClasses();
    }
```

在上面的程序，提供了如下 3 种查询方法。

1）查询指定 id 的 Student 的 Address 信息。

2）查询指定 id 的 Student 的所有 Courses 信息。

3）查询指定 id 的 Student 所属的 Classes 信息。

在进行查询时，需要按照如下流程实现。

第一步：使用 beginTransaction()方法创建一个 Hibernate 会话对象，并开始一个新 Hibernate 事务。

第二步：通过 session.load()方法获得给定 ID 的 Student 对象，实现代码如下。

```
Student st=(Student)session.load(Student.class,id);
```

第三步：调用 student.getXXX()方法返回指定的对象。

3．删除数据

如果表的关系比较复杂，则需要删除一些不需要的数据。假设要删除某个指定 id 的 student 对象的所有相关的记录，实现删除数据的代码如下。

```
/**
 *删除和某个学生相关的所有信息
 *（这里只是测试，暂且不说这种操作的意义何在）。
 */
public void delteStudent(String id)throws HibernateException
{
    beginTransaction();
    Student st=(Student)session.load(Student.class,id);
    Address addr=(Address)session.load(Address.class,st.getId());
    //删除 address 信息。
    session.delete(addr);
    //删除 classes 信息。
    session.delete(st.getClasses());
    /**
     *逐个删除 course。
     */
    for(Iterator it=st.getCourses().iterator();it.hasNext();)
    {
        Course c=(Course)it.next();
        session.delete(c);
    }
    //最后，删除 student 对象。
    session.delete(st);
    endTransaction(true);
}
```

在执行删除前，首先使用 beginTransaction()方法创建一个新 Hibernate 会话和一个新 Hibernate 事务，然后把要删除的对象装载进来，接下来调用 session.delete()方法来删除指定对象。如果要删除的是集合中的对象，那么可以通过一个迭代来逐个删除，如上面的代码中删除 courses 的方法。

19.5　疑难问题解析

在本章的内容中，详细讲解了 Hibernate 框架的基本知识。本节中，将对本章中比较难理解的问题进行讲解。

读者疑问：在开发程序的过程中，需要将 Hibernate 转换成数据库中的表，请问如何完

成操作？

解答：这是一个十分简单的问题，用户只需要配置 hibernate.cfg.xml 文件即可，具体方法为选择 myeplise 中的 Configure 选择卡，在 Properties 段中单击 “Add” 按钮，在下拉列表属性中选择

hbm2ddl.auto

值可以写 create/update

意思是根据.hbm.xml 生成数据库 DDL，即生成表。

如果在源码模式下：

<session-factory>

节点下添加以下内容：

<property name="hbm2ddl.auto">update</property>

读者疑问：在使用 Hibernate 的过程中，总是出现错误，提示信息为 org.hibernate.InvalidMapping-Exception:Could not parse mapping document from invalid mapping 什么地方有问题？

解答：重新配置*.hbm.xml 文件，一般来说，是 xml 中的字段与数据库表的字段有冲突，删除表测试一下，也可以在 hiberntate.cfg.xml 里面设置了<property name="hbm2ddl.auto"> create</property>即可解决。



职场点拨——谈升职

一个程序员最需要具备什么？技术当然是主要的，但是团队合作精神也必不可少。大多数 HR 经理很注重 Team build 的 Team leader。好的项目经理不会总是把责任推给下属，好的下属也不会总是不满意而经常炒老板。在 IT 公司内，一个典型项目团队的结构如图 19-17 所示。



图 19-17　项目团队的结构

每一个层级的工资水平自然也是不一样的，跨越一个层级的待遇就会不同。一般情况如下。

1）实习程序员是最基层的，流动性也是最大，待遇最低，没有任何职称。

2）程序员也是基层的，区别在于是正式员工，享受正式员工的基本待遇，有职称，是

“程序员”。

3）软件工程师，是我们这个部门开发的中坚力量，会带领一部分程序员开发，也可能是归属于其他项目中。他们是比较优秀的程序员晋升上来的。

4）开发项目经理，负责项目质量，项目进度，项目开发的技术把关。

上述是一条主流的升职路线，当然最终还不止是开发项目经理。

2009 年 XX 月 XX 日，阳光明媚

EE 公司考核人员，在名单上赫然列出了小菜和小李两个人的名字，并有简单的分析总结如下。

1）小李一直在做开发，有两年的开发经验，为人也不错，与世无争的一种感觉，基本上领导委派的工作他都能很快完成，不是自己的分内的事情就不去想了，对新知识的敏感度不强。

2）小菜在公司工作两年，对系统的开发也非常熟悉，但是对未知事物非常敏感，而且自己非常愿意钻研一些新的技能，他对 Linux 操作系统比较熟悉，平时爱钻研这方面的开发技巧。



其实平时对软件工程师是没有 Linux 系统开发要求的，但是，这时候正好有一个 Linux 系统的移植任务，最终 HR 选择了小菜。

2009 年 XX 月 XX 日，阳光明媚

今天是一个值得高兴的日子，因为我收到了一封升职 E-mail，正式升为软件工程师。我感觉到今天的天气真好，心情爽，本来以为会给人际关系最好的小李的，没想到是我，赶紧电话通知媳妇准备出去吃一顿……

关于为什么让小菜升职，HR 只说了一句话：“你虽然年轻，但是有上进心，而小李与世无争，无上进心。”

上述升职经历只是个例，但是也说明了一些问题：上进心比与世无争更重要。现实中每天都会上演升职故事，我们都很希望成为其中的一员。下面给出 5 条程序员升职技巧，希望对读者有所帮助。

（1）学无止境

即使你是一位拥有 10 年以上开发经验的程序员，也需要使劲学习。因为计算机是一个充满创造力的领域，新技术的更新频率大大超过了其他领域。要想跟上时代的步伐，我们必须去了解新的程序语言，了解正在发展中的程序语言，以及一些编程框架并且关注业内的新闻，及时把握整个软件开发的趋势。

（2）使用版本管理工具管理代码

资深程序员都会使用源码的版本管理，大多数开发团队都有版本管理系统。使用什么样的版本管理工具依赖于开发团队的大小和地理分布，作为一个优秀的程序员，使用一款源码版本管理工具来管理自己的代码是很有必要的。

（3）必须是一个优秀的团队成员

在今天，几乎没有一个成熟的软件是一个人独立完成的。你可能是团队中最牛的程序员，但是技术好并不意味着是好的团队成员。请问你在和你的团队成员交流中有礼貌吗？你

是否经常和他们沟通，并且大家都喜欢和你在一起讨论问题？团队是项目开发的单位，一个合理团结的团队才能开发出优秀的项目。

（4）及时准备备份和注意安全

丢失东西是永远无法避免的，例如，你的笔记本电脑被人偷了，你的硬盘损坏了，你的计算机中病毒了，你的系统被人入侵了，等等。所以，做好备份工作是非常非常重要的事情。硬盘是不可信的，要定期刻录光盘。网络也是不可信的，要随时预防病毒和黑客的侵袭。

第 20 章　Struts 2、Spring 和 Hibernate 整合应用

对于 Struts 2、Spring 和 Hibernate 这 3 个框架来说，它们各有各的优点，三者之间完美的配合能够实现 Java Web 的强大功能，可以开发出十分优秀的程序。本章将介绍使用这 3 个框架实现整合开发的过程。

20.1　效果展示

在本书前面的三章内容中学习了 Struts 2、Spring 和 Hibernate 这 3 个框架，这 3 个框架结合起来才更有魅力，例如，Spring 和 Hibernnate 整合，Struts 2、Spring 和 Hibernate 整合。本章实例能够实现一个注册功能，如图 20-1 所示的是需要的注册页面。

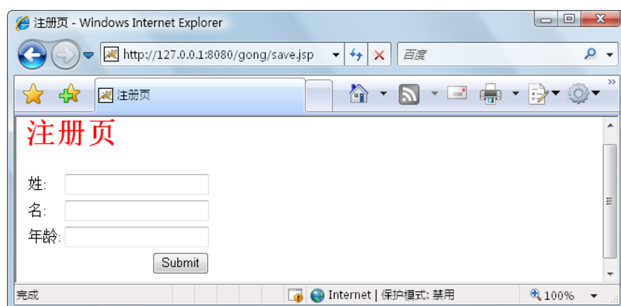


图 20-1　注册的页面

通过注册后，用户可以查看数据库的内容，并可以对它进行更改或删除，效果如图 20-2 所示。

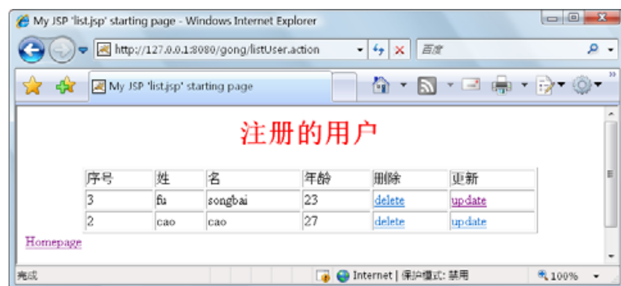


图 20-2　最终结果

20.2　搭建一个整合环境

要开发一个 Struts 2、Hibernate 和 Spring 的工程，必须先搭建一个环境，只有这样，才能顺利开发工程，下面将逐一讲解开发环境的配置。

20.2.1　Tomcat 虚拟目录与 MyEclipse 一致

在开发工程前，首先要建立一个工程的文件，如在 E 盘的 gong 文件夹建立一个 san 工程，然后将 Tomcat 的虚拟目录进行配置，以方便调试，其具体操作如下。

1）启动 MyEclipse，将工程指向 E 盘，然后新建一个名为“san”的工程，并选择“MyEclipse/Preferences”命令，如图 20-3 所示。

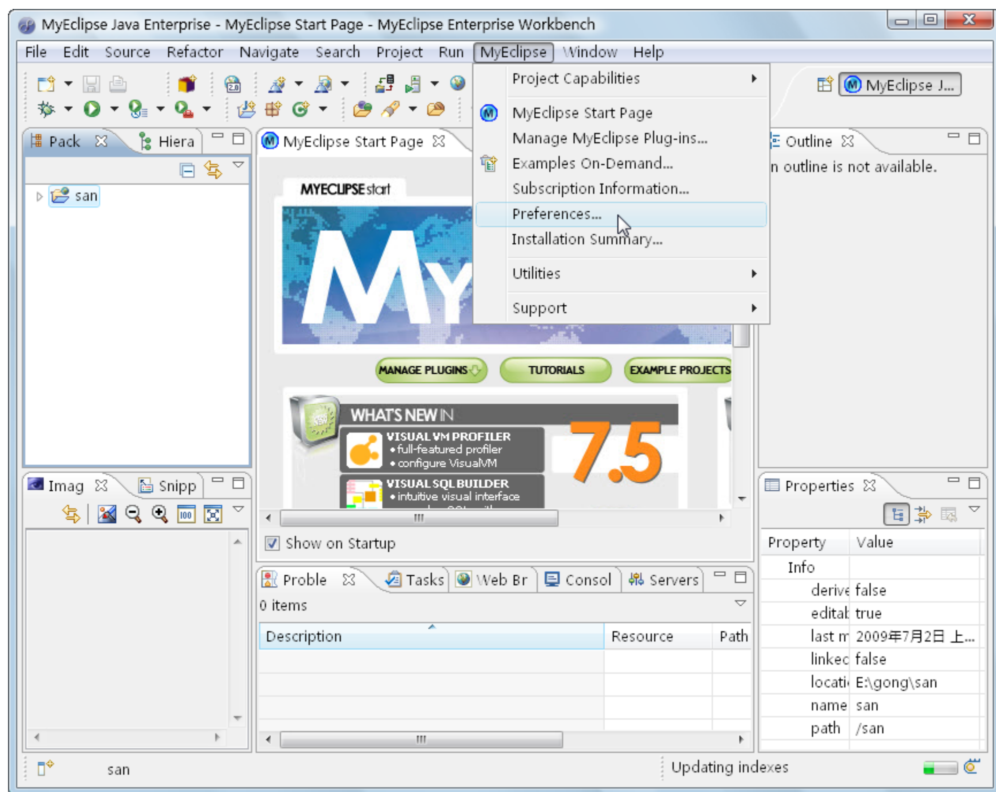


图 20-3　选择命令

2）在打开的窗口中的左边栏中选择“MyEclipse Enterprise Workbench”，依次选择下面的“Server”选项、“Tomcat”选项，然后再选择 Tomcat 6.x，如图 20-4 所示。

3）指定 Tomcat 的安装位置，如这里是 C:\Program Files\Java\Tomcat 6.0，则指定后的界面效果如图 20-5 所示。

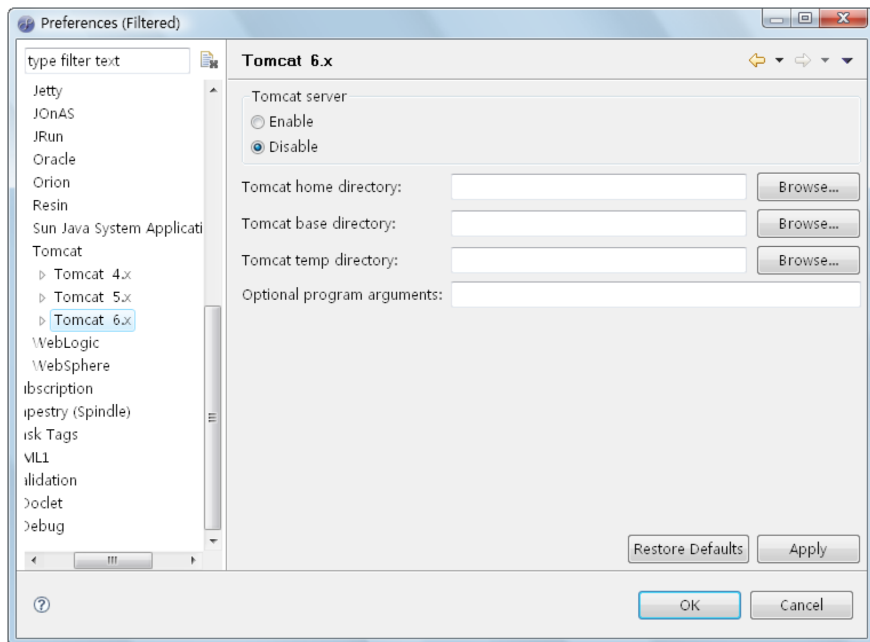


图 20-4　设置 Tomcat

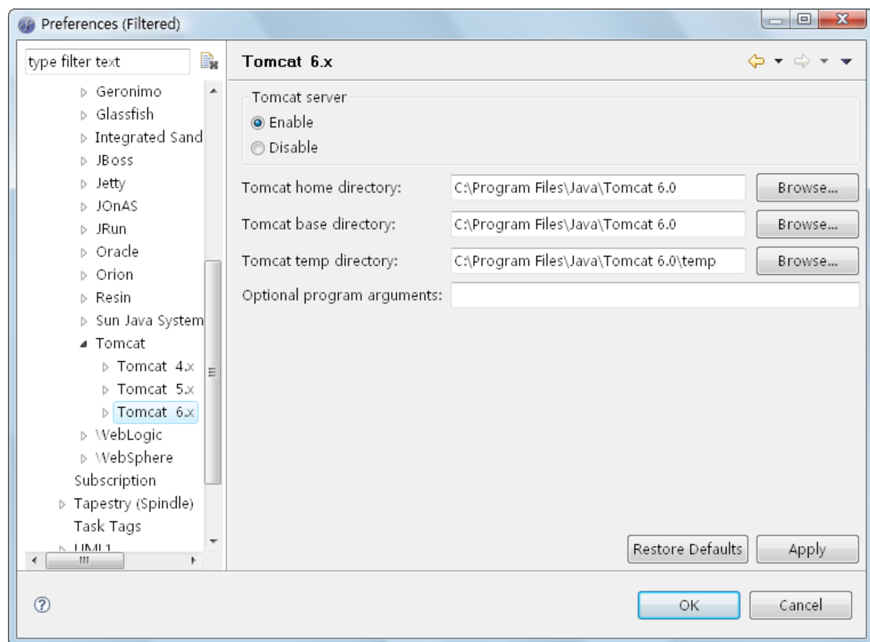


图 20-5　指定 Tomcat 的安装位置

4）单击左边的 Tomcat 6.x，选择 JDK，然后单击右上角的“Add”按钮，指定 JDK 的位置，然后单击“Finish”按钮，如图 20-6 所示。

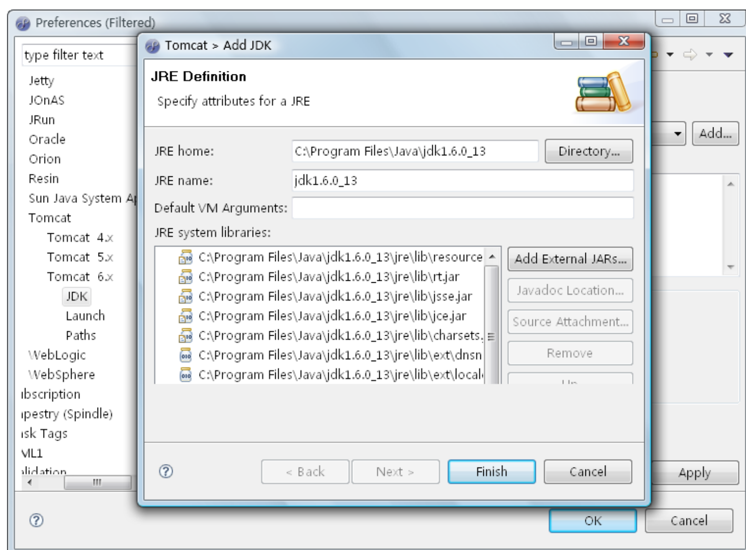


图 20-6　指定 JDK 的位置

提示：在 MyEclipse 中自带了 Java 虚拟机，但是因为版本过老，不方便调试，所以开发者一般都不使用它。开发人员一般使用最新的 JDK，以便更好地兼容 Struts 2、Hibernate 和 Spring 的整合开发。

5）单击左边的 Launch 选项，然后在右边选择“Run mode”单选按钮，单击“OK”按钮，如图 20-7 所示。

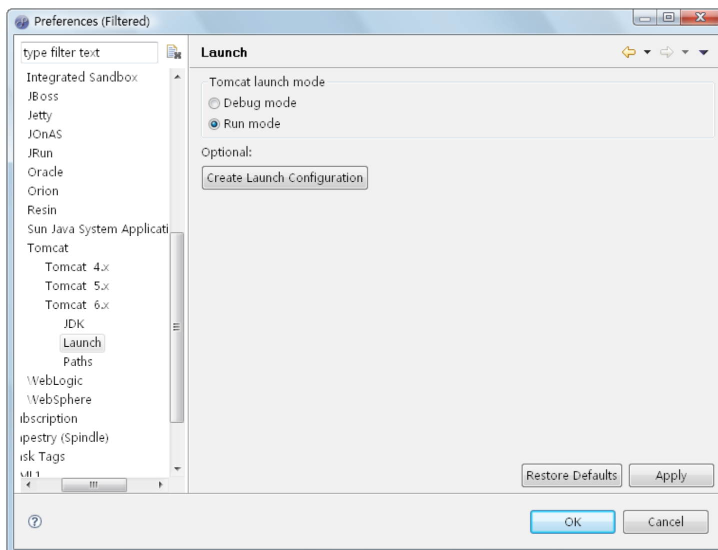


图 20-7　设置 Launch

6）回到 MyEclipse 开发界面，单击常用工具栏中的按钮，打开“Project Deployments”对话框，然后单击“Add”按钮，为“san”这个项目添加一个 Web 服务器，如图 20-8 所示。

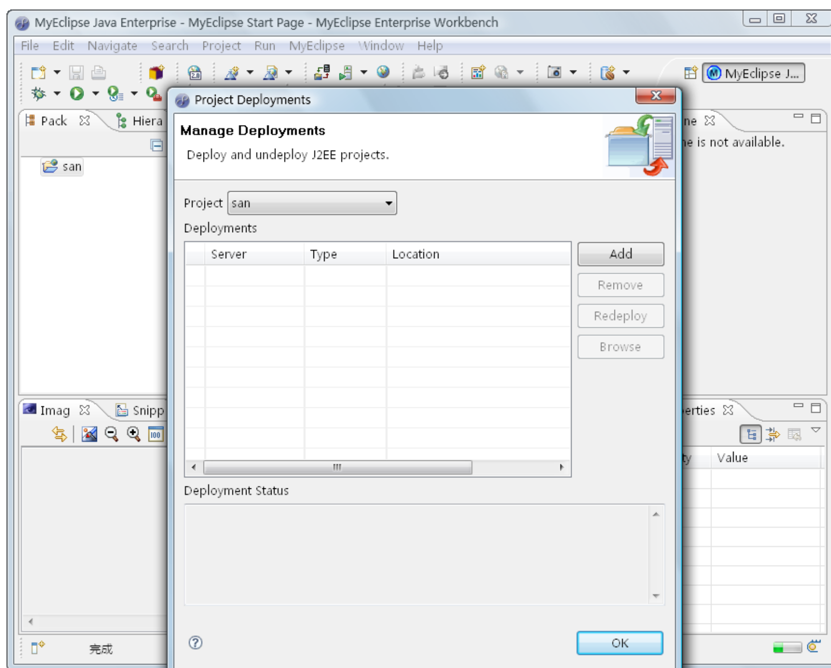


图 20-8　为项目添加服务器

7）在打开的对话框中的 Server 下拉列表中选择“Tomcat 6.x”选项，选择后，单击“Finish”按钮，如图 20-9 所示。

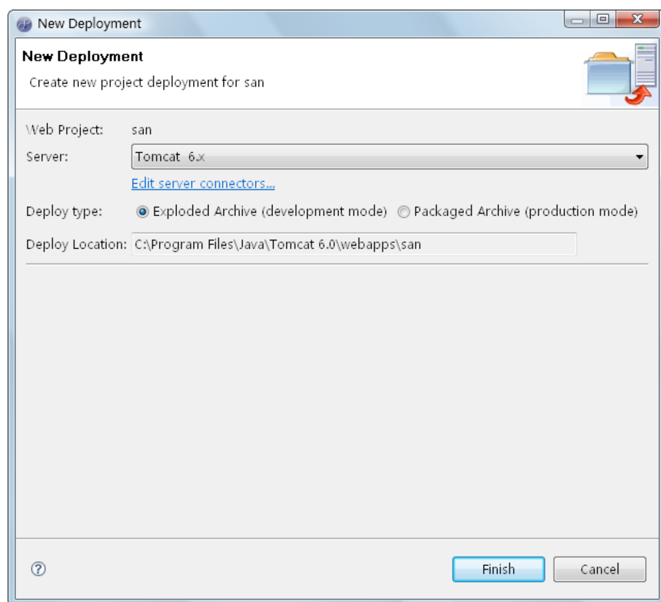


图 20-9　选择服务器

8）用文本编辑器打开 Tomcat 下的 conf 下的 Server.xml 文件，修改虚拟目录，如 C:\Program Files\Java\Tomcat 6.0\conf\下的 Server.xml，然后在</host>标签中，指定工程的虚拟目录，<Context path="/gong" docBase="E:\gong\san\WebRoot" debug="0" reloadable="true"/>，如

图 20-10 所示。

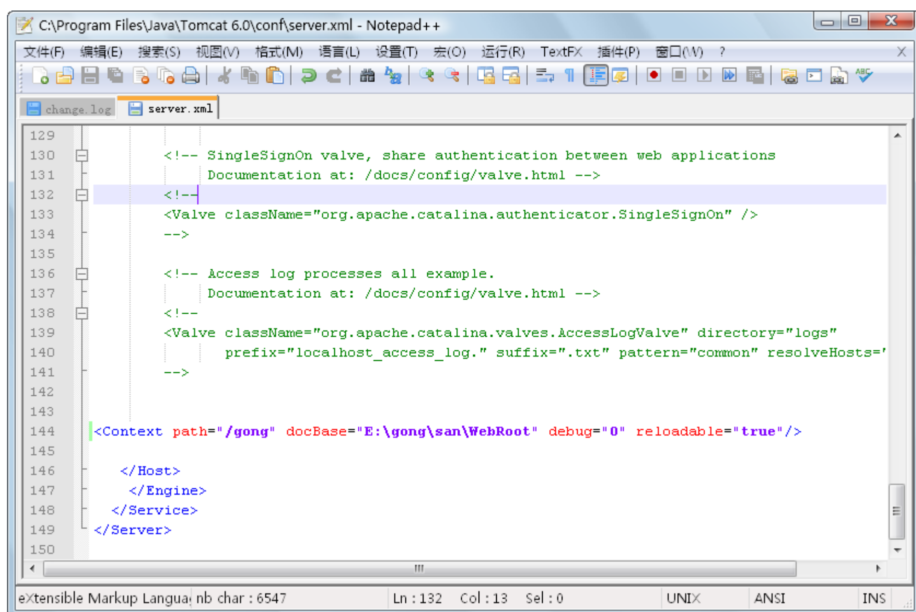


图 20-10 指定虚拟目录

9）在 MyEclipse 中启动 Tomcat 服务器，单击常用工具栏中的按钮，在弹出的窗口中选择“Tomcat/Start”命令，如图 20-11 所示。

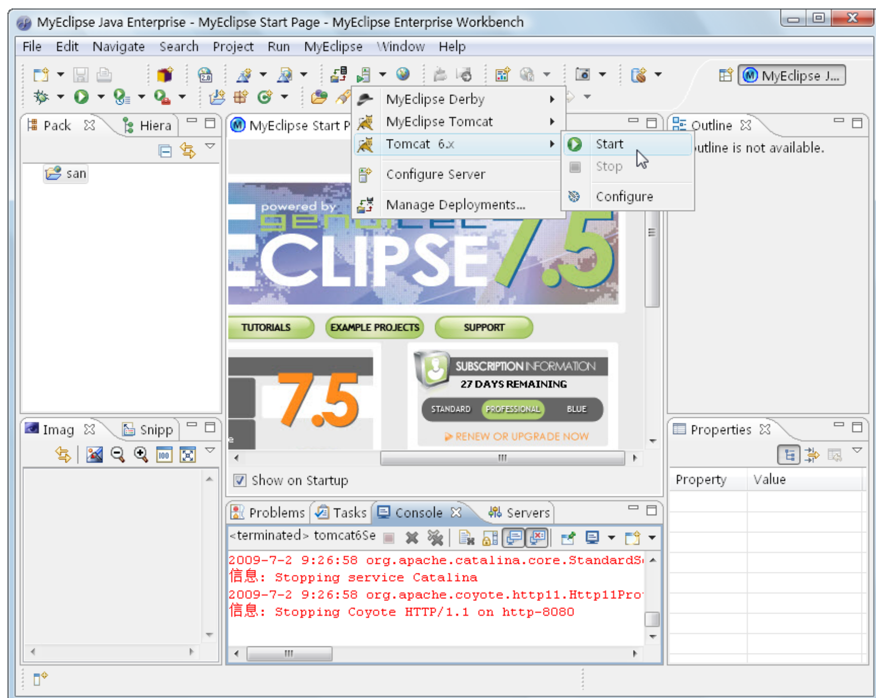


图 20-11 启动 Tomcat

10）在浏览器中输入测试地址 http://127.0.0.1:8080/gong/后，将得到成功的测试页面，如图 20-12 所示。

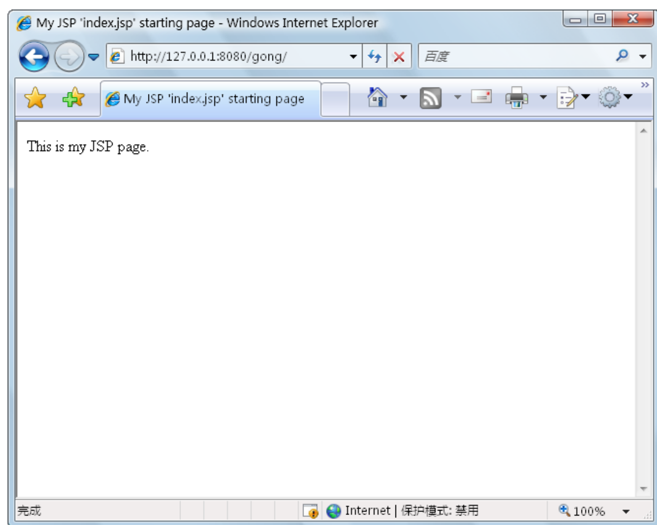


图 20-12　测试页面

20.2.2　先搭建 Hibernate

Hibernate 用户不必去官方网站下载文件，用户可以直接使用 MyEclipse 提供的 Hibernate 来建立工程，其具体操作流程如下。

1）选择左边的工程名，依次选择“MyEclipse/Project Capabilities/Add Hibernate Capabilities”命令，如图 20-13 所示。

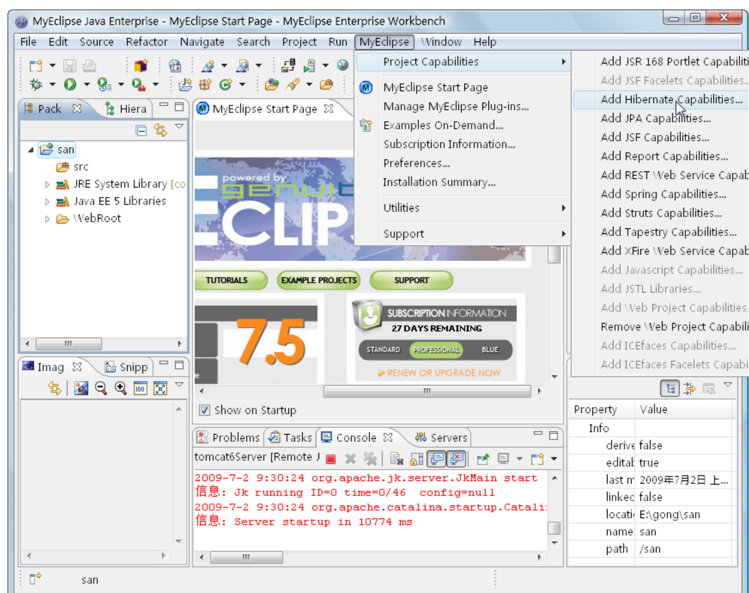


图 20-13　选择命令

2）在打开的对话框中选择 hibernate 3.2 版本，具体设置如图 20-14 所示，设置完成后单击“Next”按钮。

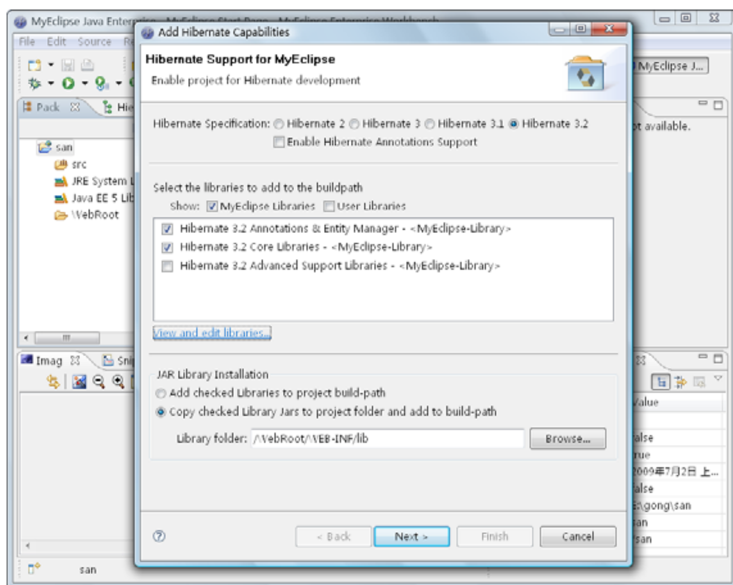


图 20-14　设置 hibernate

3）在打开的对话框中保持默认设置，单击“Next 按钮”，在打开新的对话框中设置数据库的连接信息，在此先不配置，取消☑ Specify database connection details? 复选框，如图 20-15 所示。

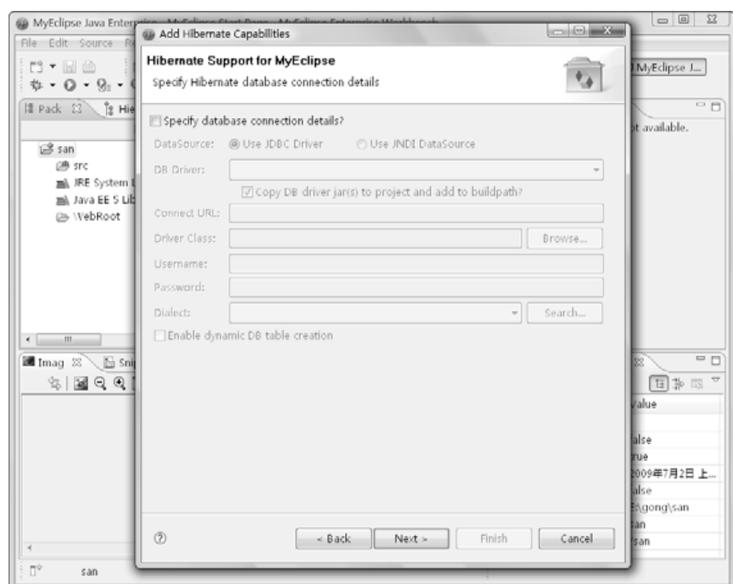


图 20-15　不配置数据库信息

4）创建一个实例，取消☑ Create SessionFactory class? 复选框，单击“Finish”按钮，如图 20-16 所示。

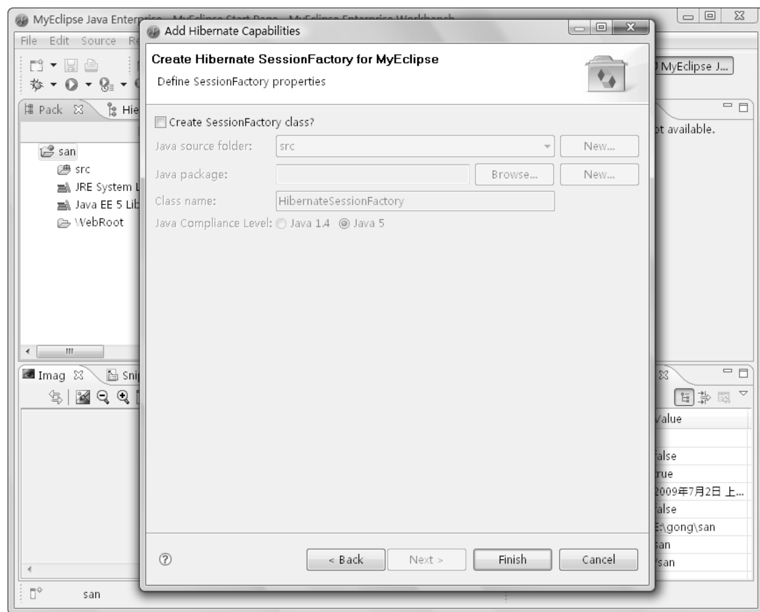


图 20-16　不创建实例

提示：在配置 Hibernate 的向导里，不使用 Hibernate 创建实例和数据库信息，主要是将一些功能让给 Spring 框架处理。

5）搭建成功后，会打开如图 20-17 所示的效果图。

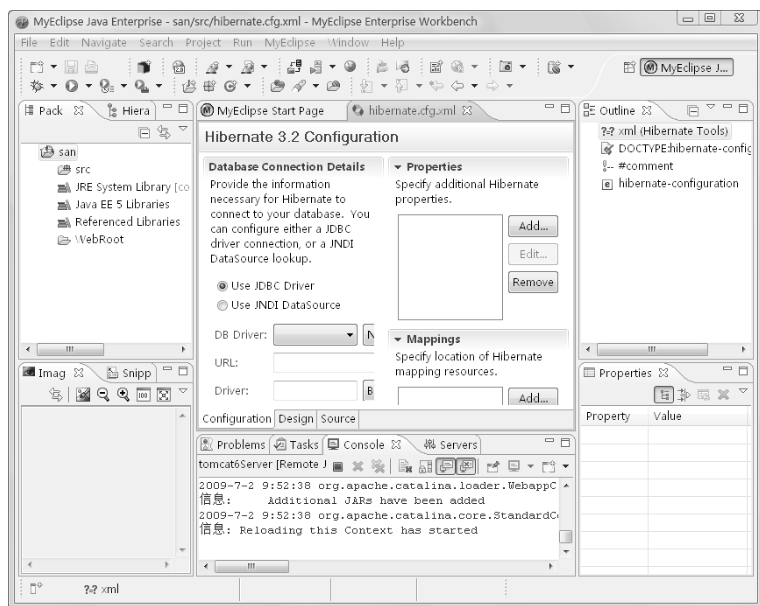


图 20-17　Hibernate 的创建

20.2.3　再搭建 Spring

搭建好 Hibernate 后，用户不妨将 Spring 的运行环境搭建好。搭建 Spring 十分简单。读

者也可用 MyEclipse 中自带的 Spring，不必去官方网站下载或者自助搭建。

1）选择左边的工程名，依次选择“MyEclipse/Project Capabilities/Add Spring Capabilities”命令，如图 20-18 所示。

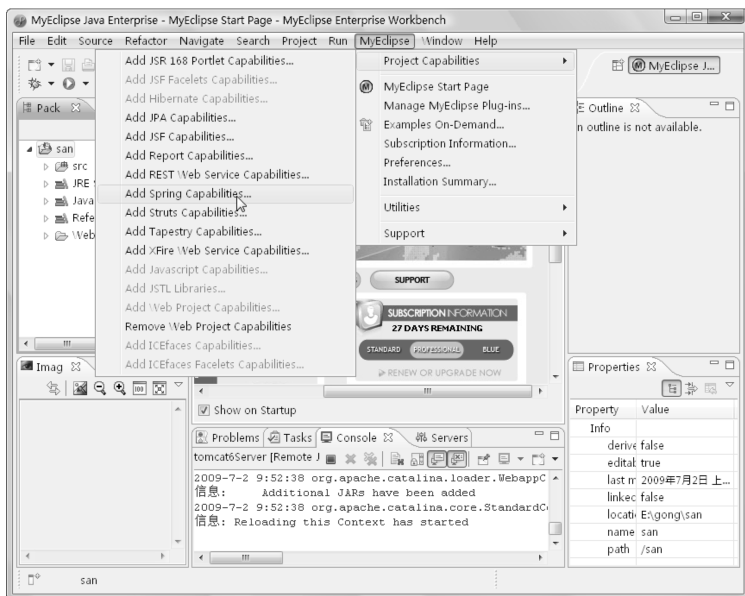


图 20-18　选择 Spring 命令

2）然后选择一个 Spring，在此选择 Spring 2.0，在下面的 Jar 选项卡中，首先选择前 4 个复选框，然后选择 Spring 2.0 Web Libraries 复选框，其他设置如图 20-19 所示。

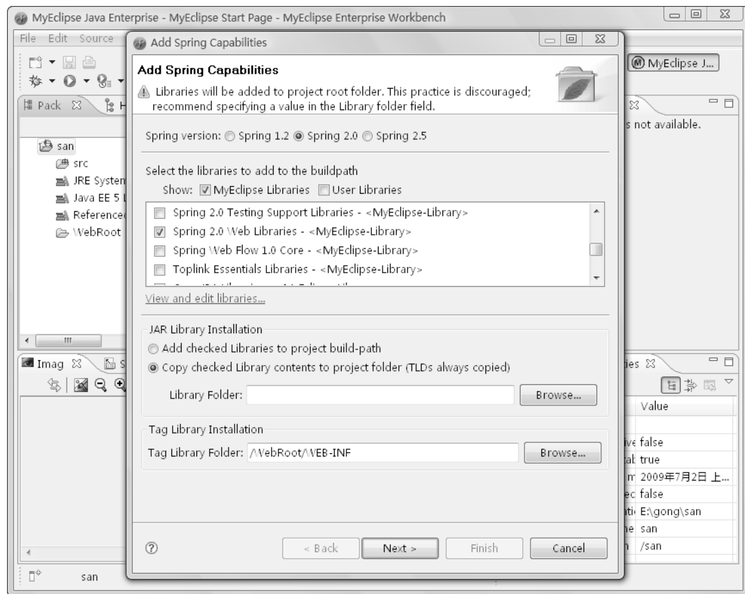


图 20-19　设置 Spring

3）在打开的对话框中，取消 Enable AOP Builder 复选框，然后将路径指定到 WebRoot\WEB-INF 下，设置好后，单击“Next”按钮，如图 20-20 所示。

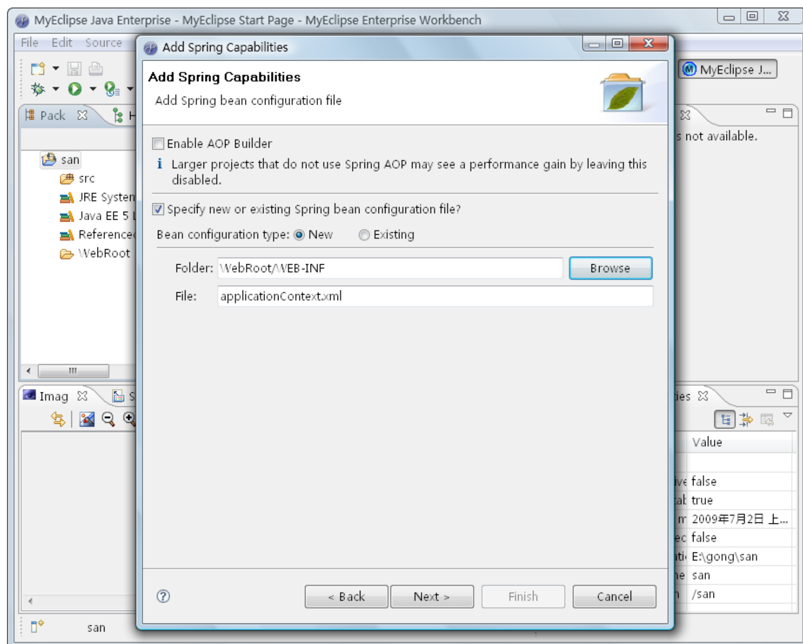


图 20-20 设置 Spring

4）取消所有的复选框，Spring 设置希望通过手工完成，不需要它自动生成的代码，设置完成后单击“Finish”按钮，如图 20-21 所示。

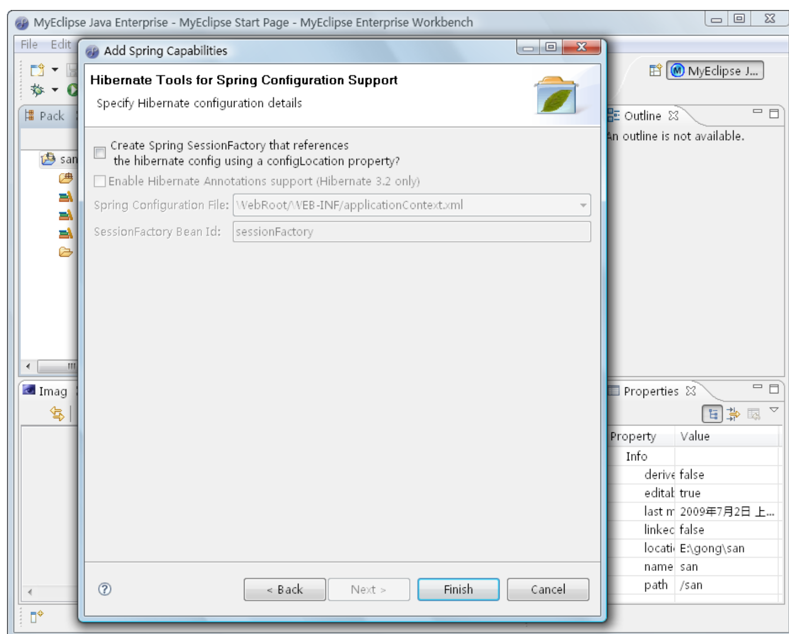


图 20-21 设置 Spring 是否自动生成配置

5）完成后打开 Spring 的配置文件，这是一个空的 Bean 文件，如图 20-22 所示。

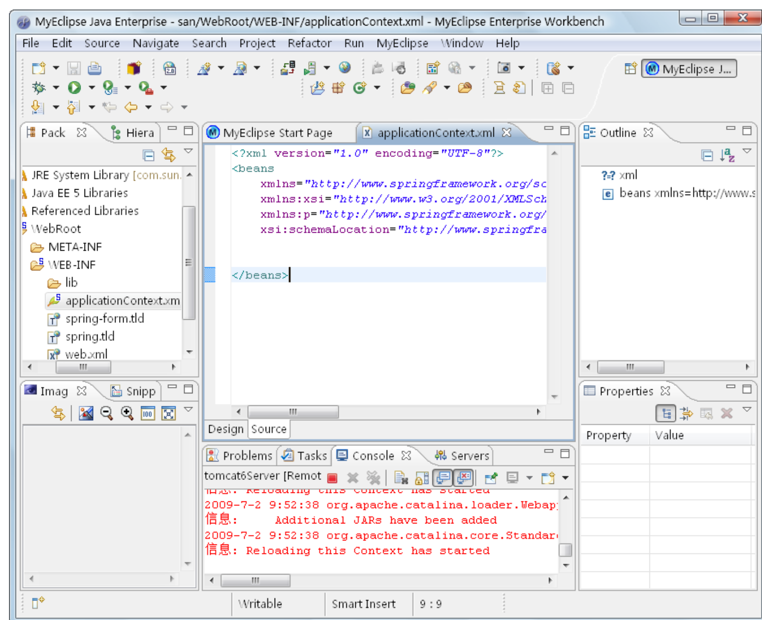


图 20-22　Spring 成功配置

20.2.4　Struts 2 的添加

在前面曾经讲过 Struts 的增加，在目前的 MyEclipse 中，并不支持 Struts 2，用户需要自动添加。读者可以从 Struts 的官方网站下载，然后将需要的包添加进去，其具体操作流程如下。

1）在下载的 Struts 文件的 lib 文件夹中选择需要的 jar 文件，在此需要选择 4 个文件，分别是 freemarker-2.3.13.jar、ognl-2.6.11.jar、struts2-core-2.1.6.jar 和 xwork-2.1.2.jar，还需要选择一个很重要的包，Struts 2 整合 Spring 的包 struts2-spring-plugin-2.1.6.jar，如图 20-23 所示。

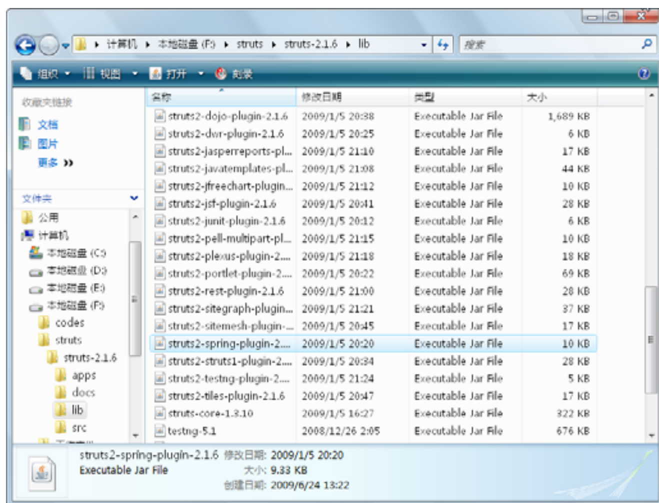


图 20-23　选择需要的 Jar 文件

2）将上述 5 个 jar 文件复制到项目中的 WEB-INF 的 Web 文件夹，如图 20-24 所示。

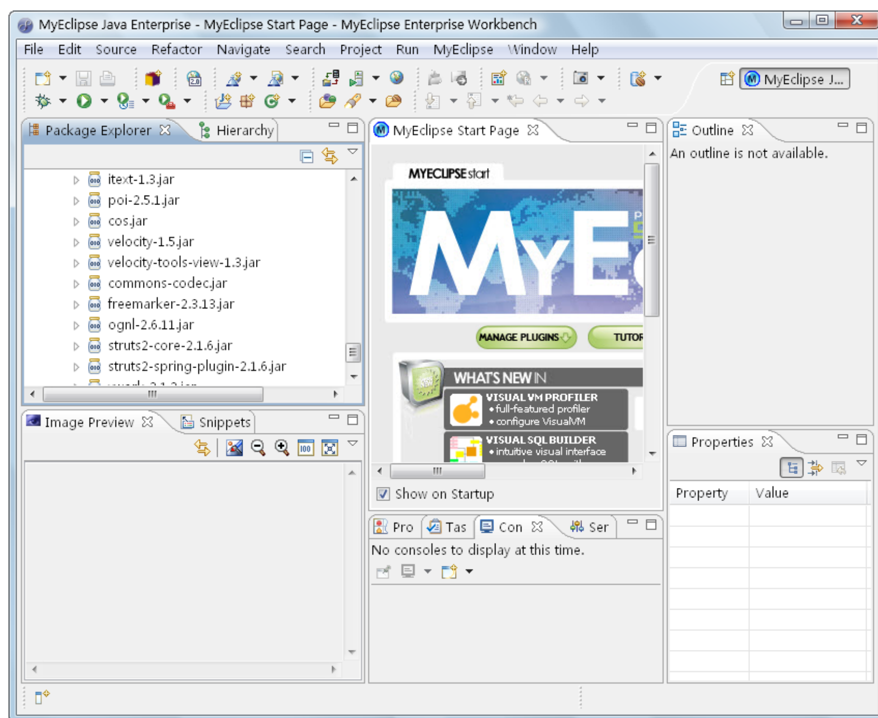


图 20-24　添加 Struts 2 的包

3）新建一个 Struts 包，然后配置文件 web.xml，其配置代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
   http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

   <filter>
         <filter-name>struts2</filter-name>
         <filter-class>
               org.apache.struts2.dispatcher.FilterDispatcher
         </filter-class>

   </filter>
   <filter-mapping>
         <filter-name>struts2</filter-name>
         <url-pattern>/*</url-pattern>
   </filter-mapping>

   <welcome-file-list>
         <welcome-file>index.jsp</welcome-file>
```

```
        </welcome-file-list>

        <listener>
             <listener-class>
                  org.springframework.web.context.ContextLoaderListener
             </listener-class>
        </listener>
   </web-app>
```

要想 struts 2 能够正常运行，需要配置两个文件，一个是位于工程里的 src 下面的 struts.xml，这个文件添加了 struts 2 的许多功能；一个是 WEB-INF 里的 Lib 下的 web.xml，这里需要配置网站的基本功能。

20.3 简单的工程

在 Java EE 的工程中，编写得最多的是 Java 源文件和 XML 的配置文件，其 JSP 代码相对比较简单，Struts 2、Hibernate、Spring 的包都是需要 XML 文件进行配置的。

20.3.1 新建数据库

在建立数据库之前，需要将 MySQL 的 JDBC 添加到工程中去，在本书前面的内容中已经讲过，在此不再赘述。在此新建立一个 mytest 数据库，然后新建一张表，建成的数据库表如图 20-25 所示。

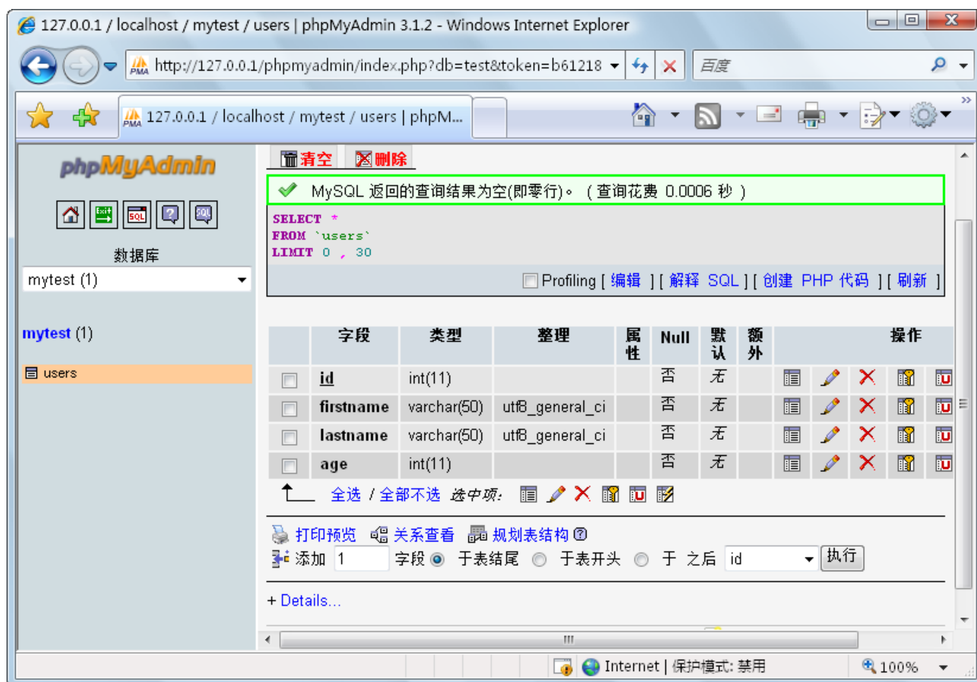


图 20-25　新建的数据库表

20.3.2　配置 Hibernate、Spring 里的信息

要想系统能正常运行，需要在 Hibernate 里配置信息，它是直接和数据库打交道的层，用户需要打开配置文件，输入如下代码。

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<!-- Generated by MyEclipse Hibernate Tools.                -->
<hibernate-configuration>
   <session-factory>
      </session-factory>
</hibernate-configuration>
```

在 Hibernate 中并没有配置什么信息，因为要将数据库里的内容交给 Spring 处理，所以需要打开 Spring 文件，并输入如下代码。

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
  xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
destroy-method="close">
  <property name="driverClassName" value="com.mysql.jdbc.Driver"></property>
  <property name="url" value="jdbc:mysql://localhost:3306/mytest"></property>
  <property name="username" value="root"></property>
  <property name="password" value="1234"></property>
  <property name="maxActive" value="100"></property>
  <property name="maxIdle" value="30"></property>
  <property name="maxWait" value="500"></property>
  <property name="defaultAutoCommit" value="true"></property>
</bean>
<bean id="sessionFactory" class="org.springframework.orm.hibernate3.
LocalSessionFactoryBean">
  <property name="dataSource" ref="dataSource"></property>
  <property name="hibernateProperties">
       <props>
            <prop key="hibernate.dialect">org.hibernate.dialect.
            MySQLDialect</prop>
            <prop key="hibernate.show_sql">true</prop>
       </props>
  </property>
  <property name="mappingResources">
```

```
            <list>
                 <value>com/test/bean/User.hbm.xml</value>
            </list>
        </property>
    </bean>
    <bean id="userDao" class="com.test.dao.impl.UserDAOImpl" scope="singleton">
        <property name="sessionFactory">
            <ref bean="sessionFactory"/>
        </property>
    </bean>
    <bean id="userService" class="com.test.service.impl.UserServiceImpl">
        <property name="userDao" ref="userDao"></property>
    </bean>
    <bean id="saveUserAction" class="com.test.action.user.SaveUserAction"
    scope="prototype">
        <property name="service" ref="userService"></property>
    </bean>
    <bean id="listUserAction" class="com.test.action.user.ListUserAction"
    scope="prototype">
        <property name="service" ref="userService"></property>
    </bean>
    <bean id="removeUserAction" class="com.test.action.user.RemoveUserAction"
    scope="prototype">
        <property name="service" ref="userService"></property>
    </bean>
    <bean id="updatePUserAction" class="com.test.action.user.UpdatePUserAction"
    scope="prototype">
        <property name="service" ref="userService"></property>
    </bean>
    <bean id="updateUserAction" class="com.test.action.user.UpdateUserAction"
    scope="prototype">
        <property name="service" ref="userService"></property>
    </bean>
    <bean id="generateExcelAction" class="com.test.action.user.GenerateExcelAction"
    scope="singleton">
        <property name="service" ref="userService"></property>
    </bean>
    </beans>
```

20.3.3 新建几个页面

JSP 页面十分简单，一般是用来显示信息和输入信息。接下来建几个基础页面和配置文件，index.jsp 十分简单，只有两个链接，在此就不展示代码了。其中有一个 save 页面，用来向数据库输入信息，里面的标签是 struts 里的标准标签，其代码如下。

```
    <%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
```

```
<%@ taglib prefix="s" uri="/struts-tags"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
      <title>注册页</title>
     </head>
   <body>
      <h1><font color="red">注册页</font></h1>
       <s:form action="saveUser">
        <s:textfield name="user.firstname" label="%{getText('firstname')}">
</s:textfield>
        <s:textfield name="user.lastname" label="%{getText('lastname')}">
</s:textfield>
        <s:textfield name="user.age" label="%{getText('age')}"></s:textfield>
        <s:submit></s:submit>
            </s:form>
   </body>
</html>
```

上面的页面是一个注册信息页面，对它进行处理需要申请一个类，用文件 user.java 来处理上面这个页面的信息，具体代码如下。

```
package com.test.bean;
public class User
{
   private Integer id;
   private String firstname;
   private String lastname;
   private int age;
   public Integer getId()
   {
         return id;
   }
   public void setId(Integer id)
   {
         this.id = id;
   }
   public String getFirstname()
   {
         return firstname;
   }
   public void setFirstname(String firstname)
   {
         this.firstname = firstname;
   }
   public String getLastname()
   {
```

```
            return lastname;
        }
        public void setLastname(String lastname)
        {
            this.lastname = lastname;
        }
        public int getAge()
        {
            return age;
        }
        public void setAge(int age)
        {
            this.age = age;
        }
    }
```

除了这个 Java 文件，还需要建立 xml 文件来处理信息，新建文件 User.hbm.xml，这个文件实际上就是 Hibernate 的映射，具体代码如下。

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="com.test.bean.User" table="users">
        <id name="id" type="java.lang.Integer" column="id">
            <generator class="increment"></generator>
        </id>
        <property name="firstname" type="string" column="firstname"
            length="50">
        </property>
        <property name="lastname" type="string" column="lastname"
            length="50">
        </property>
        <property name="age" type="java.lang.Integer" column="age"></property>
    </class>
</hibernate-mapping>
```

新建文件 SaveUserAction.java，用来保存存入数据库的文件，其代码如下。

```
package com.test.action.user;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;
import com.opensymphony.xwork2.ActionSupport;
import com.test.bean.User;
import com.test.service.UserService;
public class SaveUserAction extends ActionSupport{
    private User user;
```

```
        private UserService service;
        public User getUser()
        {
            return user;
        }
        public void setUser(User user)
    {
            this.user = user;
        }
        public UserService getService()
    {
            return service;
        }
        public void setService(UserService service)
        {
            this.service = service;
        }
        @Override
        public String execute() throws Exception
        {
            this.service.save(this.user);
            return SUCCESS;
        }
        @Override
        @SuppressWarnings("unchecked")
        public void validate()
        {
            Map map = this.getFieldErrors();
            Set set = map.keySet();
            for (Iterator iter = set.iterator(); iter.hasNext();)
            {
                System.out.println(map.get(iter.next()));
            }
        }
    }
```

20.3.4　提交页面的处理

当浏览者输入信息后，需要对数据进行处理，此数据处理功能需要交给 Dao 层进行处理，因此用户需要建立一个接口，即新建一个实现类，新建一个名为 UserDao.java，其实现代码如下。

```
    package com.test.dao;
    import java.util.List;
    import com.test.bean.User;
    public interface UserDAO
```

```
{
    public void saveUser(User user);//生成的方法
    public void removeUser(User user);//删除
    public User findUserById(Integer id);//查找
    public List<User> findAllUsers();
    public void updateUser(User user); //更新
}
```

通过上述接口写了一个实现类，通过里面的方法实现具体功能。新建文件 UserDAOImpl.java，具体代码如下。

```
package com.test.dao.impl;
import java.util.List;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
import com.test.bean.User;
import com.test.dao.UserDAO;
public class UserDAOImpl extends HibernateDaoSupport implements UserDAO{ //查询
    @SuppressWarnings("unchecked")
    public List<User> findAllUsers(){
          String hql = "from User user order by user.id desc";
          return (List<User>)this.getHibernateTemplate().find(hql);
    }
    public User findUserById(Integer id){
         User  user=(User)this.getHibernateTemplate().get(User.class,id);
          return user;
    }
    public void removeUser(User user)
    {
          this.getHibernateTemplate().delete(user);
    }
    public void saveUser(User user)
    {
          this.getHibernateTemplate().save(user);//
    }
    public void updateUser(User user)
    {
          this.getHibernateTemplate().update(user);
    }
}
```

上述文件不能直接被控制层应用，需要编写一个服务层。在服务层中定义一个接口，用来完成业务逻辑。编写文件 UserService.java，其实现代码如下。

```
package com.test.service;
import java.io.InputStream;
import java.util.List;
import com.test.bean.User;
```

```
public interface UserService
{
  public List<User> findAll();//显示用户
  public void save(User user);//保存用户
  public void delete(User user);//删除用户
  public User findById(Integer id);//返回一个而用户
  public void update(User user);//更新用户信息
  public InputStream getInputStream();
}
```

上述代码申请了服务层的接口，在此需要定义一个类，实现这个接口的方法。新建文件 UserServiceImpl.java 实现上面定义的接口，其主要代码如下。

```
package com.test.service.impl;
public class UserServiceImpl implements UserService{
  private UserDAO userDao;
  public UserDAO getUserDao(){
      return userDao;
  }
  public void setUserDao(UserDAO userDao)   {
      this.userDao = userDao;
  }
  public void delete(User user)   {
      this.userDao.removeUser(user);
  }
  public List<User> findAll()     {
      return this.userDao.findAllUsers();
  }
  public User findById(Integer id)     {
      return this.userDao.findUserById(id);
  }
  public void save(User user)     {
      this.userDao.saveUser(user);
  }
  public void update(User user)   {
      this.userDao.updateUser(user);
  }
  public InputStream getInputStream()  {
      HSSFWorkbook wb = new HSSFWorkbook();
      HSSFSheet sheet = wb.createSheet("sheet1");
      HSSFRow row = sheet.createRow(0);
      HSSFCell cell = row.createCell((short) 0);
      cell.setEncoding(HSSFCell.ENCODING_UTF_16);
      cell.setCellValue("序号");
      cell = row.createCell((short) 1);
      cell.setEncoding(HSSFCell.ENCODING_UTF_16);
      cell.setCellValue("姓");
```

```
            cell = row.createCell((short) 2);
            cell.setEncoding(HSSFCell.ENCODING_UTF_16);
            cell.setCellValue("名");
            cell = row.createCell((short) 3);
            cell.setEncoding(HSSFCell.ENCODING_UTF_16);
            cell.setCellValue("年龄");
            List<User> list = this.findAll();
            for (int i = 0; i < list.size(); ++i){
                User user = list.get(i);
                row = sheet.createRow(i + 1);
            cell = row.createCell((short) 0);
                cell.setEncoding(HSSFCell.ENCODING_UTF_16);
                cell.setCellValue(i + 1);
                cell = row.createCell((short) 1);
                cell.setEncoding(HSSFCell.ENCODING_UTF_16);
                cell.setCellValue(user.getFirstname());
                cell = row.createCell((short) 2);
                cell.setEncoding(HSSFCell.ENCODING_UTF_16);
                cell.setCellValue(user.getLastname());
                cell = row.createCell((short) 3);
                cell.setEncoding(HSSFCell.ENCODING_UTF_16);
                cell.setCellValue(user.getAge());
            }
            File file = new File("test.xls");
            try
            {
                OutputStream os = new FileOutputStream(file);
                wb.write(os);
                os.close();
            }
            catch (Exception e)
            {
                e.printStackTrace();
            }
            InputStream is = null;
            try
            {
                is = new FileInputStream(file);
            }
            catch (FileNotFoundException e)
            {
                e.printStackTrace();
            }
            return is;
        }
    }
```

上述代码十分简单，通过此类的方法可以直接访问 Dao 层。

此时上面的代码还不能运行，因为还没有配置文件 struts.xml。配置过程很简单，具体代码如下。

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
   "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
   "http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>
   <package name="user" extends="struts-default">

        <action name="saveUser" class="saveUserAction">
             <result name="success" type="redirect">listUser.action</result>
             <result name="input">/save.jsp</result>
        </action>
                  <action name="listUser" class="listUserAction">
             <result>/list.jsp</result>
        </action>
                  <action name="deleteUser" class="removeUserAction">
             <result name="success" type="redirect">listUser.action</result>
        </action>
                  <action name="updatePUser" class="updatePUserAction">
             <result name="success">/update.jsp</result>
        </action>
                  <action name="updateUser" class="updateUserAction">
             <result name="success" type="redirect">listUser.action</result>
             <result name="input">/update.jsp</result>
        </action>
                  <action name="generateExcel" class="generateExcelAction">
             <result name="success" type="stream">
                  <param name="contentType">application/vnd.ms-excel</param>
                  <param name="contentDisposition">filename=
                  "AllUsers.xls"</param>
                  <param name="inputName">downloadFile</param>
             </result>
        </action>
             </package>
</struts>
```

配置完成 struts.xml 文件后，还需要配置 Spring 文件，在前面的内容中已经讲解了配置过程，读者直接运行这个程序即可。

20.3.5　将用户信息显示出来

通过上面的代码，虽然可以将数据写入数据库，但是不能显示给浏览者。接下来建立一

个显示用户信息的类，新建文件 ListUserAction.java，其代码如下。

```
package com.test.action.user;
import java.util.Map;
import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.ActionSupport;
import com.test.service.UserService;
public class ListUserAction extends ActionSupport
{
   private UserService service;
   public UserService getService()
   {
        return service;
   }
   public void setService(UserService service)
   {
        this.service = service;
   }
   @SuppressWarnings("unchecked")
   @Override
   public String execute() throws Exception
   {
        Map request = (Map) ActionContext.getContext().get("request");
        request.put("list", service.findAll());
        return SUCCESS;
   }
}
```

新建一个类后，新建文件 list.jsp，用来显示用户信息，其代码如下。

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
      <title>My JSP 'list.jsp' starting page</title>
   <meta http-equiv="pragma" content="no-cache">
   <meta http-equiv="cache-control" content="no-cache">
   <meta http-equiv="expires" content="0">
   <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
   <meta http-equiv="description" content="This is my page">
   <!--
   <link rel="stylesheet" type="text/css" href="styles.css">
   -->
   <script type="text/javascript">
   function del()
   {
```

```
            if(confirm("你真的想删除该记录么？"))
            {
                 return true;
            }
            return false;
       }
       </script>
      </head>
      <body>

       <h1><font color="red"><center>注册的用户<br></center></font></h1>
         <table border="1" width="80%" align="center">
            <tr>
            <td>序号</td><td>姓</td><td>名</td><td>年龄</td><td>删除</td><td>更新</td>
                      </tr>
            <s:iterator value="#request.list" id="us">
            <tr>
                 <td><s:property value="#us.id"/>                </td>
                 <td><s:property value="#us.firstname"/></td>
                 <td><s:property value="#us.lastname"/></td>
               <td><s:property value="#us.age"/></td>
       <td><s:a href="deleteUser.action?user.id=%{#us.id}" onclick="return
       del();">delete</s:a></td>
    <td><s:a href="updatePUser.action?user.id=%{#us.id}">update</s:a></td>

            </tr>
       </s:iterator>
          </table>
         <s:a href="index.jsp">Homepage</s:a><br><br>
       <s:a href="generateExcel.action">生成 excel</s:a>
        </body>
    </html>
```

20.3.6　使用 struts 2 校验框架对表单进行校验

接下来需要对注册页面进行校验，看表单中的代码是不是符合要求，此时需要检验源文件。新建文件 SaveUserAction-validation.xml，其文件的代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE validators PUBLIC "-//OpenSymphony Group//XWork Validator
1.0.2//EN" "http://www.opensymphony.com/xwork/xwork-validator-1.0.2.dtd">
<validators>
   <field name="user">
        <field-validator type="visitor">
             <param name="context">user</param>
             <param name="appendPrefix">true</param>
```

```
            <message>用户信息的 </message>
        </field-validator>
    </field>
</validators>
```

然后新建一个 User-user-validation.xml，其代码如下。

```
 <?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE validators PUBLIC "-//OpenSymphony Group//XWork Validator
1.0.2//EN" "http://www.opensymphony.com/xwork/xwork-validator-1.0.2.dtd">
<validators>
  <field name="firstname">
        <field-validator type="requiredstring">
            <message>姓不能为空</message>
        </field-validator>
  </field>
  <field name="lastname">
        <field-validator type="requiredstring">
            <message>名不能为空</message>
        </field-validator>
  </field>
        <field name="age">
        <field-validator type="required">
            <message>年龄不能为空</message>
        </field-validator>
        <field-validator type="int">
            <param name="min">1</param>
            <param name="max">100</param>
            <message>年龄在 1 到 100 之间的整数</message>
        </field-validator>
  </field>
  </validators>
```

20.3.7 删除用户信息

在实际应用中，用户肯定会删除输入的信息，删除方法十分简单，首先取出值，然后将其删除，其整个实现过程如下。

新建文件 RemoveUserAction.java，其实现代码如下。

```
package com.test.action.user;
import com.opensymphony.xwork2.ActionSupport;
import com.test.bean.User;
import com.test.service.UserService;
public class RemoveUserAction extends ActionSupport
{
  private User user;
```

```
        private UserService service;

        public User getUser()
        {
              return user;
        }
        public void setUser(User user)
        {
              this.user = user;
        }
        public UserService getService() {
              return service;
        }
        public void setService(UserService service)     {
              this.service = service;
        }
              @Override
        public String execute() throws Exception   {
              this.service.delete(user);
                          return SUCCESS;
        }
        }
```

然后在 list 中的 head 标签中定义一个 JavaScript 代码，具体代码如下。

```
        <script type="text/javascript">
        function del()  {
              if(confirm("你真的想删除该记录么？")){
                    return true;
              }
              return false;   }
        </script>
```

经过上面的整理，就可以实现删除功能了。

20.3.8　更改用户信息

除了删除用户操作外，还需要更改用户的信息。接下来新建文件 update.jsp，具体代码如下。

```
     <%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
     <%@ taglib prefix="s" uri="/struts-tags" %>
     <%
     String path = request.getContextPath();
     String basePath = request.getScheme()+"://"+request.getServerName()+":
     "+request.getServerPort()+path+"/";
     %>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <base href="<%=basePath%>">
       <title>My JSP 'update.jsp' starting page</title>
   <meta http-equiv="pragma" content="no-cache">
   <meta http-equiv="cache-control" content="no-cache">
   <meta http-equiv="expires" content="0">
   <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
   <meta http-equiv="description" content="This is my page">
   <!--
   <link rel="stylesheet" type="text/css" href="styles.css">
   -->
  </head>
    <body>
        <h1><font color="red">Update User</font></h1>
      <s:form action="updateUser">
         <table>
              <tr>
                   <td><s:hidden name="user.id"value="%{user.id}">
                   </s:hidden></td>
              </tr>
                        <tr>
  <td><s:textfield name="user.firstname" value="%{user.firstname}"
label="%{getText('firstname')}"></s:textfield>
                   </td>
              </tr>
                        <tr>
                   <td>
<s:textfield name="user.lastname" value="%{user.lastname}" label="%{getText
('lastname')}"></s:textfield>
                   </td>
              </tr>
                        <tr>
 <td>   <s:textfield name="user.age" value="%{user.age}" label="%
{getText('age')}"></s:textfield>
                   </td>
              </tr>
                        <tr>
                   <td>
                        <s:submit></s:submit>
                   </td></tr>
         </table>
       </s:form>
      </body>
</html
```

新建文件 UpdatePUserAction.java，具体代码如下。

```
package com.test.action.user;
import com.opensymphony.xwork2.ActionSupport;
import com.test.bean.User;
import com.test.service.UserService;
public class UpdatePUserAction extends ActionSupport{
  private User user;
  private UserService service;
  public User getUser(){
       return user;    }
  public void setUser(User user)  {
       this.user = user;     }
  public UserService getService(){
       return service;
  }
  public void setService(UserService service)     {
       this.service = service;    }
       @Override
  public String execute() throws Exception   {
       user = this.service.findById(user.getId());
                  return SUCCESS;
  }
}
```

新建文件 UpdateUserAction.java，其代码如下。

```
package com.test.action.user;
import com.opensymphony.xwork2.ActionSupport;
import com.test.bean.User;
import com.test.service.UserService;
public class UpdateUserAction extends ActionSupport{
  private User user;
  private UserService service;
  public User getUser(){
       return user;    }
  public void setUser(User user){
       this.user = user;     }
  public UserService getService(){
       return service; }
  public void setService(UserService service)     {
       this.service = service;    }
       @Override
  public String execute() throws Exception   {
       this.service.update(user);
                  return SUCCESS;
  }}
```

在更新 JSP 文件时需要校验，其校验代码和前面类似，新建文件 UpdateUserAction-validation.xml，具体代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE validators PUBLIC "-//OpenSymphony Group//XWork Validator
1.0.2//EN" "http://www.opensymphony.com/xwork/xwork-validator-1.0.2.dtd">
<!--
<validators>
   <field name="user.firstname">
         <field-validator type="requiredstring">
               <message>姓不能为空</message>
         </field-validator>
   </field>
         <field name="user.lastname">
         <field-validator type="requiredstring">
               <message>名不能为空</message>
         </field-validator>
   </field>
   <field name="user.age">
         <field-validator type="required">
               <message>年龄不能为空</message>
         </field-validator>
         <field-validator type="int">
               <param name="min">1</param>
               <param name="max">100</param>
               <message>年龄在 1 到 100 之间的整数</message>
         </field-validator>
   </field>
</validators>
 -->
<validators>
   <field name="user">
         <field-validator type="visitor">
               <param name="context">user</param>
               <param name="appendPrefix">true</param>
               <message>user's </message>
         </field-validator>
   </field>
</validators>
```

ISBN 978-7-111-39087-9

在线互动交流平台

官方微博：http://weibo.com/cmpjsj
豆　瓣　网：http://site.douban.com/139085/
读者信箱：cmp_itbook@163.com

《C语言编程新手自学手册》
《Java编程新手自学手册》
《Java Web编程新手自学手册》
《PHP编程新手自学手册》
《Visual C++编程新手自学手册》
《C＃编程新手自学手册》
《编程算法新手自学手册》

地址：北京市百万庄大街22号
邮政编码：100037
电话服务
社服务中心：010-88361066
销售一部：010-68326294
销售二部：010-88379649
读者购书热线：010-88379203
网络服务
教材网：http://www.cmpedu.com
机工官网：http://www.cmpbook.com
机工官博：http://weibo.com/cmp1952
封面无防伪标均为盗版

上架指导 计算机/程序设计

ISBN 978-7-111-39087-9
策划编辑◎丁　诚　杨　硕/封面设计◎道乐文化



定价：89.00元